Reinhold, Paul

Identifikation markanter Spezifika von Rich Internet Applications sowie die Evaluierung deren Auswirkung auf die Entwicklung von Informationssystemen.

eingereicht als

Bachelorarbeit

an der

Fachbereich Mathematik/Physik/Informatik

Mittweida, 31.08.2009

Erstprüfer:	Prof. Dr.-Ing. Wilfried Schubert
Zweitprüfer:	Dipl.-Kaufmann Uni. Stefan Papenfuß

Bibliographische Beschreibung:

Reinhold, Paul: Identifikation markanter Spezifika von Rich Internet Applications sowie die Evaluierung deren Auswirkung auf die Entwicklung von Informationssystemen. - 2009. - 80 S. Mittweida, Hochschule Mittweida (FH), Fakultät Mathematik/Physik/Informatik, Bachelorarbeit, 2009

Referat:

Ziel der Bachelorarbeit ist es, die Anforderungen an die Entwicklung von Rich Internet Applikationen, im Vergleich zu traditionellen Web Anwendungen untersuchen und einen Überblick über die Veränderungen zu bieten. Dazu werden sowohl Merkmale von Web-Applikationen, als auch von Rich Internet Applikationen beschrieben. Ebenfalls ist eine Analyse verbreiteter Rich Internet Applications-Technologien unter den Gesichtspunkten der spezifischen Merkmale Bestandteil dieser Arbeit, um zum Abschluss die Auswirkungen auf die Entwicklung konkret beschreiben zu können.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Einführung und Problemstellung

Bei der Entwicklung von Applikationen im Word Wide Web stehen den Entwicklern verschiedene Alternativen zur Verfügung. Eine verbreitete Variante ist der Einsatz von serverseitigen Programmiersprachen, wie in Java verfasste Servlets, JavaServer Pages oder PHP, um HTML Seiten mit dynamischen Inhalt zu generieren. Die Darstellung dieser Seiten übernimmt danach ein Web-Browser. Jedoch sehen sich solche traditionellen Web-Anwendungen zunehmend mit Problemen konfrontiert, denn die Interaktionsmöglichkeiten über die grafische Oberfläche, sind im Vergleich zu Desktop-Applikationen eingeschränkt. Durch diese Tatsache lassen sich beispielsweise nicht immer alle Anforderungen des Business in Form solcher Anwendungen realisieren. Auch die Integration in komplexe E-Business Systeme ist häufig mit Schwierigkeiten verbunden. Ungenügende Usability und negative Auswirkung in der Produktivität bei der Verwendung von Web-Applikationen sind nur zwei Beispiele die hier, stellvertretend für eine Reihe von Schwierigkeiten, aufgeführt werden sollen.

Ein neuer[1] Trend im Bereich von Web-Anwendungen sind Rich Internet Applications (RIA) . Literatur und Community sind sich jedoch noch nicht einig, wodurch sich eine RIA abschließend definiert. Häufig werden in diesem Zu-

[1] Streng genommen handelt es sich in den meisten Fällen um keine neuen Technologien, sondern bekannte, neu aufbereitete; ein Beispiel dafür ist AJAX.

sammenhang konkrete Technologien genannt, mit denen sich Rich Internet Applications implementieren lassen. RIAs streben nun an die Probleme, die bei der Verwendung von traditionellen Web-Anwendungen entstehen, zu löschen und zeichnen durch sich Besonderheiten aus. Diese zu identifizieren ist ein Teilaspekt, dem sich diese Arbeit widmen soll.

Geklärt werden muss weiterhin, ob es sich bei RIAs nur um *erweiterte, herkömmliche Web-Anwendungen* handelt und die Entwicklung wie bisher erfolgt. Oder ob die besonderen Merkmale dieser Art von Applikationen, so gravierende Änderungen mit sich bringen, dass neue Schwerpunkte beim Entwickeln gesetzt werden müssen. Dies soll einen weiteren Teilaspekten der Arbeit darstellen.

1.2 Zielsetzung

In Anlehnung an die Problemstellung und die definierten Thesen ergeben sich für die Arbeit die folgenden Ziele:

- Definition von Rich Internet Applications, sowie die Identifikation auffälliger, charakteristischer Merkmale.
- Analyse von Rich Internet Applications in Hinsicht auf User Interface, Entwicklung und Architektur.
- Kategorisierung und Beschreibung verbreiteter Technologien im Umfeld von Rich Internet Applikationen.
- Identifikation einer möglichen Ursache für die spezifischen Merkmale von RIAs.
- Auffinden möglicher Veränderungen bei der Entwicklung von Rich Internet Applications im Vergleich zu traditionellen Web-Anwendungen.

Schwerpunkte dieser Arbeit sind Rich Internet Applications, mit ihren spezifischen Merkmalen, sowie die Betrachtung dieser Merkmale in Bezug auf die Entwicklung von Informationssystemen. Weiterhin sollen in der Arbeit verbreitete Technologien beschrieben, jedoch in keiner Weise bewertet werden. Auch soll die Bewertung möglicher Entwicklungsumgebungen (IDE) für RIAs nicht Inhalt dieser Arbeit sein.

1.3 Struktur der Arbeit

Die Arbeit ist grob in zwei Teile aufgliedern, die bereits in der Problemstellung angedeutet wurden. Während der erste Teil verstärkt auf die Grundlagen von Web-Applikationen und Rich Internet Applications eingeht, widmet sich der zweite Teil der intensiv der Auswirkungen auf die Entwicklung dieser Applikationen. Im Folgenden soll eine kurze Zusammenfassung der Kapitel wiedergegeben werden:

Im Kapitel 2 wird die Entwicklung des Web, sowie der Begriff *Web 2.0* beschrieben. Im weiteren Verlauf wird der Begriff Web-Applikation eingeführt und auf die Besonderheiten dieser Applikationen eingegangen. Abschließend wird in diesem Kapitel die Schichtmodellarchitektur, aus Sicht von traditionellen Web-Anwendungen, erläutert.

Kapitel 3 beschäftigt sich mit den grundlegenden Begriffen zu Rich Internet Applications. Neben einer der Definition des Begriffs geht das Kapitel vor allem auf die spezifischen Merkmale ein. Weiterhin werden die RIAs kategorisiert und entsprechen den Konzepten näher vorgestellt. Als Abschluss werden Barrierefreiheit und User Experience im Zusammenhang mit Rich Internet Applications beschrieben.

Das vierte Kapitel widmet sich konkreten RIA-Technologien und beleuchtet diese mit Hilfe der im vorherigen Kapitel gewonnenen Erkenntnisse. Letzter Punkt des Kapitels stellt eine Zusammenfassung der vorgestellten Technologien dar.

Aufbauend auf die vorhergehenden Kapitel, werden im fünften Kapitel Thesen über die Auswirkung auf die Entwicklung aufgestellt und erörtert. Ein Beispiel zur Modellierung einer Präsentationsschicht rundet dieses Kapitel ab.

Zum Ende schließen sich mit Kapitel 6 ein Fazit, in Form einer finalen Zusammenfassung aller gewonnenen Erkenntnisse, und ein Ausblick an.

Zusätzlich werden jeweils zu Beginn eines neuen Kapitels dessen Inhalte in Kurzform beschrieben, um den Leser einen besseren Überblick zu verschaffen. Dabei sollen, sofern notwendig, auffällige Beziehungen zu vorhergehenden Kapiteln aufgezeigt werden.

Ebenfalls sei zu Beginn darauf hingewiesen, dass das Literaturverzeichnis kapitelweise aufgebaut und für eine bessere Übersicht jeweils in Literatur [LIT] und Quellen aus dem Internet [URL] unterteilt ist.

2
Das Web im Wandel

Dieses Kapitel soll die Entwicklung des Web, vom Web 1.0 zum Web 2.0 und die Evolution des Web, von einem Informationslieferanten hinzu einer interaktiven Plattform zum Ausführen von Applikationen, schildern. Dabei soll auf typische Konzepte von Web-Anwendungen eingegangen werden, um als Ziel am Ende des Kapitel ein gemeinsames Verständnis vom Web als Plattform für Applikationen zu haben.

2.1 Bisherige Entwicklung des Webs

Tim Berners-Lee, heutiger Direktor des *World Wide Web Consortiums* W3C , gilt mit seinem 1989 geschriebenen Bericht als Gründer des World Wide Webs. In diesem Bericht präsentierte er ein Konzept, Informationen und deren Verteilung zu realisieren und dem Verlust von Wissen vorzubeugen[1]. Da es zu dieser Zeit keine kommerziellen Lösungen für die Beseitigung dieser Probleme gab, entwickelt er dieses Konzept. Bemerkenswert ist eine Aussage innerhalb des Berichts, nach welcher er behauptet das innerhalb der nächsten 10 Jahre, neben CERN auch der Rest der Welt sich mit diesen Problemen konfrontiert sieht.

In seinem Lösungsvorschlag machte er deutlich, dass *die Methode des Spei-*

[1] Vgl. [URL01]

cherns keine Beschränkung auf den Informationsgehalt haben dürfe[2]. Genau diese Probleme konnten aber, die zu dieser Zeit üblichen hierarchischen Systeme nicht oder nur ungenügend lösen. Berner-Lee erachtete daher auch eine vernetzte Sammlung von Daten in einem komplexen, dezentralen System als die sinnvollste Lösung. Wichtigste Entscheidung war Berner-Lees Entschluss das Hypertext bzw. Hypermedia[3] Konzept als Basis für das zu entwickelnde System zu nutzen. Dabei sollten Informationen auf verteilten Dokumenten zusammengefasst und durch Querverweise (*Hyperlinks*) untereinander in Verbindung stehen.

In den folgenden Jahren entwickelte sich das Konzept rasant weiter und erlangte weltweite Bekanntheit. Nach dem von Berner-Lee eigens einwickelten Webbrowser *Nexus* erschien 1992 *Viola*. Dies war der erste Browser der neben Text und Bildern auch Grafiken anzeigen konnte[4]. Im Jahr darauf erschien mit Marc Anderes *Mosaic for X* der Browser, der dem Word Wide Web zum Durchbruch verhalf und ab 1994 unter dem Namen *Netscape Navigator* weiterentwickelt wurde[5]. Als *milestone document* wird der im April des gleichen Jahres gefasste Entschluss des CERN bezeichnet, die Technologie des World Wild Webs öffentlich und unentgeltlich für alle Nutzer zur Verfügung zu stellen.

Die Jahre 1994 - 1998 sind die eigentlichen Boom Jahre des Web. Viele Unternehmen erkannten die Bedeutung einer eigenen Webpräsenz und die damit verbundenen neuen Geschäftsmöglichkeiten. In diesem Zeitraum stieg die Anzahl der betriebenen Webserver von 623 (1994) auf rund 1.6 Millionen (1998)[6]. Der Ausgang der Bestrebungen mit E-Commerce-Initiativen Profit zu machen ist allgemein bekannt: Als die erhofften Gewinne vieler Jungfirmen ausblieben, mündete dies im Platzen der Dotcom Blase im Jahre 2001. Zu diesem Zeitpunkt waren bereits 26 Millionen Webserver in Betrieb[7].

[2] [URL01] ... the method of storage must not place its own restraints on the information

[3] Die Begriffe Hypertext und Hypermedia werden häufig synonym gebraucht. Genauer betrachtet bezeichnet Hypertext, aber nur textuelle Inhalte, während Hypermedia auch multimediale Dokumente mit einbezieht.

[4] Vgl. [URL02]

[5] Vgl. [URL03]

[6] Vgl. [URL04]

[7] Vgl. [URL04]

2.2 Web 2.0 - Die Weiterentwicklung des Webs

Das World Wide Web vor dem Platzen der Dotcom Blase, wird heute rückwirkend als Web 1.0 bezeichnet. Es handelt sich dabei um ein Retronym, welches erst mit Einführung des Begriffs *Web 2.0* aufkam. Häufig wird der Begriff Web 2.0, auf Grund einer fehlenden Definition, als reiner Marketing Begriff abgetan. Web 2.0 als reinen technischen Fortschritt des Webs zu sehen ist, ist dabei genau so falsch, wie die Technik außen vor zu lassen. Das Web 2.0 wird daher auch als Phänomen der letzen Jahre betitelt, welches eine neue (technologische) Ära des WWW einläutet.

Tim O'Reilly, Gründer und Chef des O'Reilly Verlages, versucht mit seinem Artikel *What is Web 2.0?*[8] das Konzept des Web 2.0 greifbar zu machen und identifiziert in diesem sieben Merkmale, welche für Rich Internet Application ebenfalls relevant sind.

- **Web als Plattform**
  Während des Web 1.0 war die Meinung verbreitet, dass derjenige der den Browsermarkt dominierte, auch im Web an vorderster Front mitmischt. Vor allem Microsoft und Netscape waren zwei prominente Verfechter dieser Auffassung. Wie jedoch heute bekannt ist, stellte sich diese Annahme als falsch heraus. Nicht die Browser oder Webserver generieren Wert, sondern die Services die über das Web angeboten werden. Google war eines der ersten Unternehmen welches das erkannte. Die Services werden auf Plattformen bereitgestellt und nicht als Software-Pakete verkauft.

- **Die Intelligenz des Kollektivs**
  Zahlreiche Unternehmen machen sich die Benutzer ihrer Web-Applikation zu nutze, um einen besseren Service anbieten zu können. Dabei ist oft zu beobachten, dass sich mit steigender Nutzeranzahl auch der Dienst stetig verbessert und damit der Wert des Unternehmens kontinuierlich zunimmt. O'Reilly schließt daraus, dass diese vernetzen Effekte der Nutzerbeiträge der Schlüssel zum Erfolg im Web 2.0 Zeitalter sind.

- **Daten als wichtigstes Gut**
  Wie bereits angedeutet stellen die Daten, welche zum überwiegenden Teil von den Nutzern bereit gestellt werden, im Web 2.0 Umfeld eine zentrale Rolle dar. Jedoch werden diese Daten im Gegensatz zu Desktop-

[8] Vgl. [URL05]

Applikationen nicht lokal, sondern in riesigen Online-Datenpools gespeichert.

- **Abkehr von Software Releases**
  Da im Zentrum von Web-basierter Software, Services und nicht mehr Produkte stehen, führt dies auch zu einigen grundlegenden Veränderungen in den Geschäftsmodellen der Unternehmen. So werden Applikationen nicht mehr als Pakete ausgeliefert, sondern in kleinen, inkrementellen Schritten immer weiterentwickelt. Typisch ist ebenfalls, dass die Services als *perpetual beta*[9] den Nutzern angeboten werden, wodurch die Benutzer, durch ihre Feedbacks, indirekt als Entwickler/Tester tätig sind.

- **Schlanke Programmiermodelle**
  O'Reilly beschreibt als weiteren wichtigen Baustein für Web 2.0 Anwendungen schlanke Web-Services. Er behauptet, dass das RESTful Web-Services auf Grund ihrer einfacheren Verwendung, den schwergewichtigeren, auf SOAP (Simple Object Access Protocol) basierenden, vorzuziehen sind. RSS führt er ebenfalls als einen solchen leichtgewichtigen und verbreiteten Web-Service auf.

- **Geräte-Unabhänigkeit**
  Auch wenn der Rechner weiterhin das primäre Gerät bleiben wird, mit Hilfe dessen auf Web Services zugegriffen wird, ist es zunehmend wichtiger, dass Web-Applikationen auch von anderen Endgeräten abrufbar sind (z.B. Smartphones).

- **User Experience**
  Dieser Begriff ist auch für Rich Internet Applikationen von zentraler Bedeutung. So ist es das Ziel moderner Web-Applikationen diese stets benutzerfreundlicher und einer Desktop-Applikation ähnlicher werden zulassen.

Zusammenfassend lässt sich daraus sagen, dass es sich beim Phänomen Web 2.0, um eine Plattform handelt, die auf innovativen Technologien beruht und den Nutzer in den Mittelpunkt des Geschehens rückt. Bemerkenswert ist, dass die Ideen Berner-Lees in Grundzügen denen entsprach, die auch O'Reilly in seiner Definition von Web 2.0 beschreibt. Nutzer und Daten als das wichtigste Gut und die Intelligenz des Kollektivs, sind genau die Punkte die auch zur Entstehung des World Wide Web beigetragen hatten.

[9] Unbefristete Betaversion ohne Releasedatum

2.3 Web-Applikationen

Wie beim Web 2.0 gibt es keine allgemein gültige Definition des Begriffs. Bob Baxley, als ein bekannter Autor auf diesem Gebiet, nennt jedoch hierzu mindestens zwei Eigenschaften, die für Web-Applikationen typisch sind. Als ersten Punkt führt der die nutzerspezifische Interaktion auf, meist durch Identifikation des Nutzers mit Name und Passwort. Der zweite Punkt ist das Erstellen, Bearbeiten und Löschen von Informationen durch den Nutzer und damit eine bestehende Veränderung der Daten. Dennoch existieren Web-Applikationen, die ohne eine Anmeldung auskommen und damit dem ersten Punkt widersprechen. Daher ist eine andere Aussage die er trifft besser als mögliche Definition geeignet. „Der grundlegende Zweck von Web-Applikationen ist es, eine oder mehrere Aufgaben zu vollführen.“[10] Das heißt im Unterschied zu traditionellen Webseiten, erwartet der Nutzer von einer Webapplikation also immer eine konkretes Ergebnis.

2.3.1 Typische Merkmale

Web-Applikationen unterscheiden sich in zahlreichen Punkten von herkömmlichen Desktop-Anwendungen. Dieser Abschnitt soll einige typische Merkmale von Web-Anwendungen auflisten[11]:

- **Zentrale Datenhaltung**
  Web-Applikationen greifen auf zentral gespeicherte Datenbestände zu. Dies befähigt jedem Nutzer dazu, Daten in Echtzeit zu erhalten. Anderen falls wären Anwendungen, die Buchungen oder Bestellungen über das Internet erlauben nicht möglich. Durch den Online-Zugriff auf die Daten, ist es weiterhin möglich, dass auch umfangreiche Informationssysteme aufeinander zugreifen können.

- **Distribution**
  Eine clientseitige Installation entfällt bei einer Web-Anwendung. Diese Tatsache stellt für den Nutzer eine wesentliche Erleichterung dar. Jedoch bringt diese Tatsache noch weitere Vorteile mit sich. So entfällt es ebenfalls, die Software als Ware zu verpacken und auszuliefern. Da dadurch sinken die Kosten, die üblicher Weise für die Distribution und den Installationssupport anfallen, enorm. Der Nutzer erhält nur das was er unbedingt benötigt, in diesem Falle das User Interface. Ein weiterer großer

[10] Vgl. [URL06] ... The fundamental purpose of all web applications is to facilitate the completion of one or more tasks.

[11] Vgl. [URL07], [URL08], [URL09]

Vorteil ist die Weiterentwicklung der Anwendung. Entwickler können diese permanent aktualisieren, ohne dass sich der Anwender diese Updates über Umwege zugänglich machen müsste. Das im vorherigen Abschnitt beschriebene *perpetual beta*-Prinzip wird genau dadurch ermöglicht.

- **Zugriff**
  Da sich sowohl Anwendung, als auch Daten an zentraler Stelle befinden, ist es möglich diese portabel oder mobil zu machen. Einzige Vorraussetzung zur den Zugriff, ist ein Zugang zum Internet. Bietet die Applikation nun ein Frontend für mobile Endgeräte, ist es möglich mit PDAs oder Smartphones auf die Anwendung und damit ebenfalls auf die Daten zuzugreifen. Das hat den weiteren Vorteil, dass mobile Geräte Daten unmittelbar verändern können und nicht nachträglich synchronisiert werden müssen.

- **Bedienung**
  Für die Bedienung von Web-Applikationen haben sich bereits gewisse Standards etabliert. Einige Eingabemuster wie *Home*, *Back* oder *Forward* haben sogar, auf Grund ihrer Beliebtheit, in Desktop-Applikationen Einzug gehalten. Dieser Sachverhalt trägt dabei stark zum vereinfachten Erlernen der Bedienung bei.

- **Kollaboration**
  Die Zusammenarbeit zwischen Nutzern war eine der wichtigen Möglichkeiten die das World Wide Web bieten sollte. Analog dazu bieten Web-Anwendungen neue Features wie Online-Meetings oder Online-Präsentationen, an denen mehrere Nutzer gleichzeitig teilnehmen können.

Wie an den beschrieben Merkmalen zu sehen, bieten Web-Anwendungen einige Vorteile gegenüber Desktop-Anwendungen und sind damit in einigen Anwendungsfällen nicht mehr wegzudenken. Jedoch haben sie auch Nachteile, die sie für einige Fälle ungeeignet machen. So ist Konnektivtiät ein solcher Nachteil, denn um Web-Applikationen nutzen zu können, ist eine permanente Verbindung zum Internet erforderlich. Selbst wenn bereits Web-Anwendungen mit Offline-Funktionen existieren, ist ein unerwünschter Verbindungsabbruch stets ein Problem. Die ständige Verbindung zum Internet beherbergt weitere Gefahren, wie Angriffe durch Hacker oder durch Maleware. Die für Web-Anwendungen typischen Server-Roundtrips stellen, mit dem regelmäßigen Neuladen der Seite, zudem eine deutliche Verzögerungen in der Bedienung der Applikationen dar.

Somit existieren Für und Wider für diese Art von Applikationen. Jedoch sollen Web-Applikationen, Desktop-Anwendungen auch nicht in allen Punkten übertrumpfen. John Gruber[12] beschreibt dies sehr treffend, indem er sagte, dass *Web-Applikationen herkömmliche Desktop-Applikationen [...] auf einer ganz anderen Ebene schlagen.*[13]

2.3.2 Schichtenarchitektur (Layering)

Für die erfolgreiche Entwicklung von Web-Applikationen reicht es nicht sich ausschließlich für einen Programmiersprache oder eine Technologie zu entscheiden. Ebenso bedeutend ist es einen Bauplan festzulegen, der den grundsätzlichen, logischen Aufbau der Anwendung festlegt. In der Softwareentwicklung spricht man in Zusammenhang mit Architektur oft von Schichten, Layern oder Tiers[14]. Im Folgenden sollen nun mögliche Architekturmodelle für Web-Applikationen beschrieben werden[15].

- Die 1-Schichtarchitektur beschreibt die einfachste Variante und findet typischer Weise in rein clientseitigen Applikationen, wie Tabellenkalkulationen oder Videoschnittprogrammen, Anwendung. Dieser Ansatz ist für Web-Applikationen jedoch ungeeignet, da in dieser Form nur ein System involviert ist.

- Die 2-Schichtarchitektur beschreibt das typische Client-Server-Modell. Klassisches Beispiel ist dafür die Verbindung zwischen Web-Server, der statischen HTML-Code generiert und sendet, und Web-Browser, der diesen Code interpretiert und anzeigt.

- Die 3-Schichtenarchitekur beschreibt das heute am häufigsten anzutreffende Modell. Dabei wird das 2-Schichtenmodell um eine weitere Schicht zur Datenhaltung, meist eine Datenbank, erweitert. Der AMP-Stack (Apache, MySQL, PHP) ist ein solcher Vertreter für dieses Modell.

Dabei lassen sich in Anlehnung an ihre Funktionen des jeweiligen Layers Bezeichnungen wie Präsentation, Applikation und Daten Tier bestimmen[16]. Jedoch treten auch bei einer 3-Schichtenarchitektur Probleme auf. Diese gehen

[12] Betreiber des Blogs : *Daring Fireball*

[13] Vgl. [URL10] ... web apps don't need to beat desktop apps on the same terms. What's happened is that they're beating them on an entirely different set of terms.

[14] Sowohl Schicht, Layer als auch Tier, sollen in Rahmen dieser Arbeit als Synonym verwendet werden, auch wenn es in der Literatur teils Unterschiede gemacht werden. Vgl. [URL11]

[15] Vgl. [URL12], [URL13]

[16] Vgl. [URL11]

dadurch hervor, dass keine saubere Trennung zwischen Präsentationslogik und Geschäftslogik möglich ist. Denn beide befinden sich in der zweiten Schicht, wie die folgende Abbildung 2.1 verdeutlicht.[17].

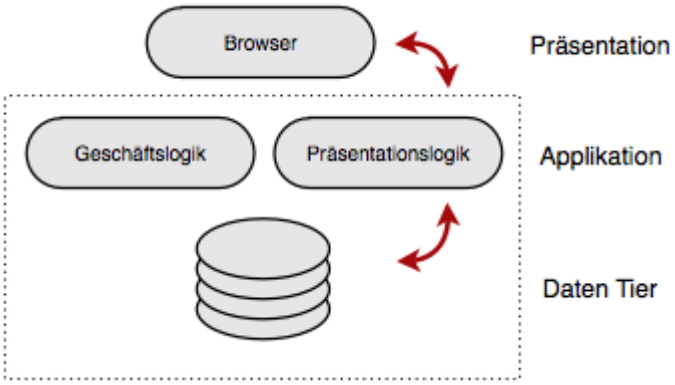

Abbildung 2.1: 3-Schichten Modell

Um das Problem zu lösen besteht die Möglichkeit weitere Schichten einzuführen, wie es bei einer n-Schichtenarchitektur der Fall ist. Abbildung 2.2 soll dieses Modell veranschaulichen.

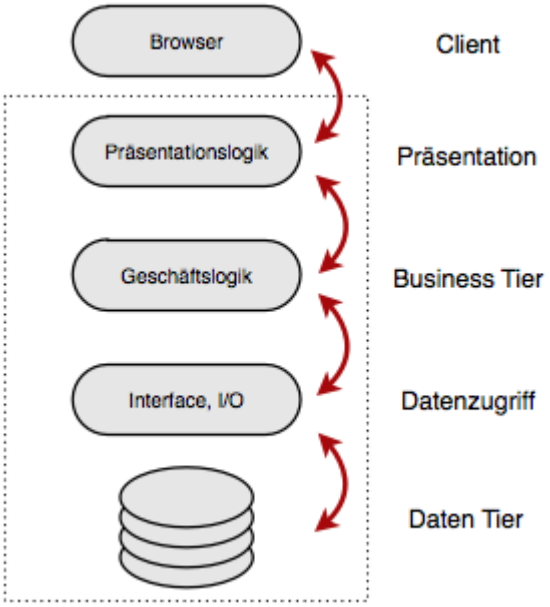

Abbildung 2.2: n-Schichten Modell

[17] Der gepunktete Rahmen ist die logische Zusammenfassung der Serverseite, d.h. eines oder mehrerer Server.

In diesem Modell wird die eben kritisierte Applikationsschicht in Präsentation und Geschäftslogik aufgeteilt, sowie eine zusätzliche Schicht für den Datenzugriff hinzugefügt. Unverändert besitzt auch im n-Schichtmodell die Datenschicht die gleichen Aufgaben, wie in der 3-Schichtarchitektur. Dabei handelt es ich meist um eine Datenbank mit dazugehörigen Datenbank-Managementsystem. Eine XML-Datei ist aber ebenfalls eine Möglichkeit für die Datenhaltung.
Die neu eingeführte Schicht des Datenzugriffs, übernimmt die Funktion einer Schnittstelle zwischen Datenhaltung und Geschäftslogik. Dabei dient sie als Vermittler zwischen diesen beiden Schichten. Objekt-relationales Mapping mit Hibernate[18], stellt eine Möglichkeit dar diese Datenzugriff zu realisieren. Diese zusätzliche Schicht den Vorteil, die Datensicht oder das DBMS ohne eine Veränderung der Geschäftslogik austauschen zu können.
Innerhalb der Geschäftslogik befinden sich die Kernfunktionalitäten der Applikation. Besonders in der Objekt-orientierten Programmierung kommt dieser Schicht nach [URL11] eine wichtige Bedeutung zu, da innerhalb der Geschäftsschicht die Objekte als Abbilder der realen Welt modelliert werden.
Die Präsentationsschicht stellt das Interface der Applikation zum Nutzer dar. Es wandelt die Ausgaben auf der Geschäftsschicht in eine Form, die den Nutzer präsentiert werden kann um. Wichtig ist, dass die Präsentationsschicht gewährleistet das Präsentationslogik und Geschäftslogik nicht vermischt werden.
Zu guter Letzt stellt die Clientschicht sicher, das die Veränderungen der Präsentationsschicht auch tatsächlich angezeigt werden. Vergleichbar ist dies mit dem Neuladen einer Webseite im Web-Browser.

Durch die Erweiterung des 3-Schicht zu einem n-Schichtmodell, ist es möglich geworden Präsentationslogik und Geschäftslogik sauber voneinander zu trennen. Auch die Einführung eine Datenzugriffsschicht trägt zu besseren Trennung der einzelnen Schichten bei, so dass die Verbindung zwischen Schichten gehalten werden kann, eine Durchmischung aber ausgeschlossen wird[19].

Auf dieses Schichtmodell wird unter 3.4 noch einmal aus der Sicht von Rich Internet Applications eingegangen. Anzumerken ist, dass das Schichtmodell vereinfacht wurde, um den Fokus auf die Besonderheiten von RIAs innerhalb der Layer zu lenken. Zudem kommt die Kommunikation als weiterer Punkt[20] hinzu, da sich die Kommunikation von traditionellen Web-Applikationen gegenüber der von RIAs maßgeblich unterscheidet.

[18] Siehe `https://www.hibernate.org/`
[19] Vgl. [URL11]
[20] Jedoch nicht als weitere Schicht

3
Rich Internet Applications

Im vorhergehenden Kapitel wurden die Entwicklung des Webs und der Web-Applikationen beschrieben, um eine Grundlage für das Verständnis der folgenden Kapitel zu erlangen. Dieses Kapitel soll nun umfassend in das Thema Rich Internet Applications einführen und dabei auf Merkmale und mögliche Konzepte eingehen.

3.1 Rich Internet Application Einführung

Joshua Duhl beschreibt den Begriff *Rich Internet Application* in einem Whitepaper im November 2003. Dabei definiert er RIA wie folgt[1]:

> ... Macromedia defines RIAs as combining the best user interface functionality of desktop software applications with the broad reach and low-cost deployment of Web applications and the best of interactive, multimedia communication. The end result: an application providing a more intuitive, responsive, and effective user experience. ...

Eine Rich Internet Application ist demnach eine Verknüpfung des Besten von Desktop-Applikationen mit dem Besten vom Web unter Verwendung von interaktiver Kommunikation. Daraus resultiert eine Applikation mit einer intuitiveren, zugänglicheren und effektiveren User Experience[2]. Das Beste der

[1] Vgl. [URL15, S. 7]
[2] Siehe 4.7 RIA und User Experience

Desktop-Applikationen bezieht sich dabei auf die Benutzerschnittstelle, die Möglichkeiten für Validierung und Formatierung, ebenso wie auf schnelle Antwortzeiten ohne Ladezeiten und desktop-übliche Aktionen wie *Drag an Drop* bietet. Das Beste vom Web beinhaltet dabei sowohl die schnelle Entwicklung als auch Plattformunabhängigkeit. Auch der fortlaufende Download von Inhalt und Daten sowie die weithin angenommenen Webstandards zählen dazu. Die zweiseitige Interaktion ist dabei Inhalt des Besten der Kommunikation.

Als Rich Internet Applications werden Applikationen oder Services im Web bezeichnet, auf die diese Merkmale zutreffen. Eine konkrete Einordnung soll weiter verdeutlichen, wie der Begriff Rich Internet Application zu verstehen ist. Als Basis dient das Web 2.0 mit Funktionen und Merkmalen wie Blogs, Social Software und der Nutzung der kollektiven Intelligenz. RIAs stellen ebenfalls ein solches Merkmal vom Web 2.0 dar. Innerhalb von Rich Internet Applications lassen sich nun wiederum typische Technologien vom Web 2.0 , wie AJAX, Flash oder Java, einordnen. RIA kann man somit als eine Zusammenfassung der technologischen Umsetzungen des Web 2.0 verstanden werden.[3] Abbildung 3.1 verdeutlicht diesen Zusammenhang noch einmal grafisch.

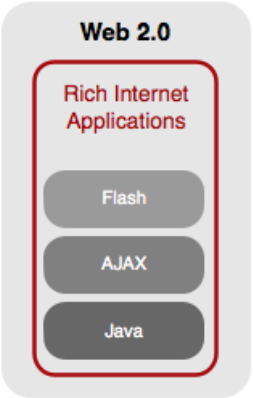

Abbildung 3.1: Zusammenhang Web 2.0 und RIA

Typisch für RIAs ist die Möglichkeit, performante Internet-Anwendungen mit nahezu uneingeschränkten Design- und Interaktionsmöglichkeiten zu realisieren, indem die lokale Rechenleistung der aufrufenden Clients genutzt wird. RIAs sind dabei typischerweise fachlich anspruchsvolle Anwendungen, mit deren Hilfe auch komplexere Abläufe übersichtlich realisiert oder unterstützt werden. Dabei soll die zur Verfügung stehende Netzwerkbandbreite bestmöglich

[3] Vgl. [URL16, S. 4]

ausgenutzt werden. Grund dafür ist, dass - vom Starten der Applikation abgesehen - nur Daten (keine Layoutinformationen) mit dem Server ausgetauscht und auf dem Client berechnet und verändert werden. Dadurch wird das Netz weniger beansprucht, als es bei ausschließlich auf dem Server stattfindenden Berechnungen der Fall ist. Die größte Performancesteigerung ergibt sich aus der Verringerung der Ladezeiten, die sich wiederum aus der Realisierung von Funktionalität auf dem Client unter optimaler Verwendung der verfügbaren Netzbandbreite ergibt. Erkauft werden diese Vorteile durch eine deutlich erhöhte Komplexität der Anwendung. Besonders erhöht sich die Komplexität einer RIA Anwendung innerhalb ihrer Präsentationsschicht. Dieses Phänomen ist bereits seit langem zu beobachten: Von den ersten einfachen Informationssystemen, über klassische formulargeführte Anwendungen, bis hin zur stark interaktiven Web 2.0 bzw. Desktop-ähnlichen Applikationen sind die Anforderungen an die Präsentationsschicht der Anwendungen immer weiter gestiegen.[4].

3.2 Spezifische Merkmale von RIAs

3.2.1 Single Page Prinzip

Herkömmliche Webanwendungen haben ein typisches, synchrones Anfrage-Antwort Verhalten: Wenn der Benutzer einen Klick auf einen Link setzt, dann wird dadurch eine Anfrage ausgelöst, die zum vollständigen Laden der angeforderten HTML Seite führt. Im Gegensatz dazu ist es erklärtes Ziel einer typischen Rich Internet Application, nur diejenigen Teile der Seite neu zu laden, die sich tatsächlich in der Zwischenzeit verändert haben. Dies muss dabei nicht synchron mit den Nutzeraktionen sein, sondern kann auch unabhängig von dessen Aktionen asynchron im Hintergrund geschehen. Auf diese Weise können zum Beispiel Eingaben in ein Textfeld eines Formularen bereits asynchron validiert werden, während der Nutzer das Formular weiter ausfüllt. So kann der Benutzer durch die Anwendung bereits während der Datenerfassung auf dem Client auf mögliche Fehler aufmerksam gemacht werden, ohne dass dabei die zu prüfenden Daten zuvor explizit an den Server gesendet werden mussten. Dadurch können auch unnötige Page Refreshes vermieden werden. Die Erfassung von Daten in Formularen kann dadurch wesentlich beschleunigt werden. Neben der reinen Validierung der Daten können dem Nutzer auch weitere Informationen vom Server bereits während der Datenerfassung auf dem Client zur Verfügung gestellt werden: Beispielsweise kann nach der korrekten

[4] Vgl. [LIT02, S. 1]

Eingabe einer Bankleitzahl auf dem Client asynchron bereits auf dem Server nach dem erfassten Namen des Bankinstituts gesucht werden, damit diese anschließend dem Nutzer zeitnah präsentiert werden kann. Dadurch erhöht sich die Benutzerfreundlichkeit der Anwendung deutlich: Der Nutzer kann sofort sehen, ob er seine Daten korrekt eingegeben hat. Ein weiterer Vorteil der asynchronen Kommunikation einer Rich Internet Application ist die bessere Ausnutzung der Bandbreite der Internetverbindung. Bei den übertragenen Daten handelt es sich nur um jene, die sich auch tatsächlich verändert haben. Dadurch wird die zu übertragende Datenmenge verringert und der Browser kann schneller diejenigen Bereiche aktualisieren, für welche er neuen Daten erhalten hat. Möglich wird diese Form der asynchronen Kommunikation durch Technologiekonzepte, wie beispielsweise AJAX. Mit Hilfe von JavaScript-Funktionen und einem so genannten XMLHttpRequest-Objekt können Daten asynchron im Hintergrund übertragen werden. Ausführlicher wird auf diese Technologie im Kapitel 4 eingegangen. Jedoch hat dieses Kommunikationsmodell auch Nachteile: Zum Beispiel müssen zu Beginn der Kommunikation deutlich mehr Daten geladen werden, als es bei herkömmlichen Webanwendungen der Fall ist. Dadurch kommt dem Nutzer der Start der Applikation unverhältnismäßig lang vor. Auf diesen Sachverhalt wird im späteren Verlauf des Abschnittes noch einmal näher eingegangen. Ein weiterer Nachteil sind die Probleme, welche die Benutzung des Zurück-Buttons im Browser auslösen kann: Beispielsweise kann der Nutzer nach Betätigen der Zurück-Schaltfläche nicht mehr nachvollziehen, welche zuvor ausgelösten Prozesse nun möglicher Weise im Hintergrund ausgeführt werden.

3.2.2 Modellierung von RIAs

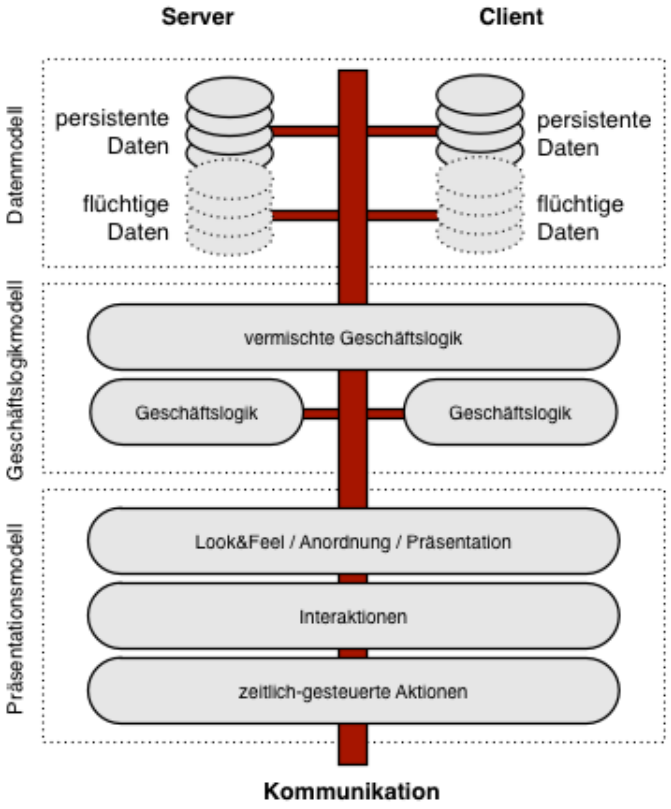

Abbildung 3.2: Modell - Rich Internet Applications

Das folgende Modell soll einen Überblick vermitteln, wie die unter 2.3.2 beschriebenen Sichten im RIA - Umfeld miteinander agieren. Dazu wird im Folgenden auf jede der umrahmten Modelle und auf die Kommunikation, als Verbindung zwischen den Schichten, eingegangen.

3.2.3 Datenmodell

Die Möglichkeiten zur Verteilung von Daten auf Client und Server sind bei Web-Applikationen stark limitiert. Üblicherweise befindet sich bei datenintensiven Applikationen der überwiegende Teil der Daten auf der Serverseite. Die Speicherung von Information auf der Clientseite in Form von Cookies ist stark eingeschränkt. Durch Rich Internet Applications steht nun auch der clientseitige Speicher verstärkt zur Verfügung. Seine Verfügbarkeit ist jedoch abhängig von den Einstellungen des Clients und von der jeweiligen verwendeten Technologie. Zum Beispiel können ein Einkaufswagen einer e-Commerce Applikation

oder ein Terminkalender einer Kalender-Anwendung lokal gespeichert werden. In einer RIA werden die Daten auch auf dem Client bearbeitet. Zum Beispiel können bereits auf dem Client bestellte Waren einem Einkaufswagen hinzugefügt bzw. diesem wieder entnommen werden. Nach Speichern und Bearbeiten von Daten auf dem Client werden diese Daten gemeinsam am Ende des Anwendungsfalls innerhalb einer einzigen Transaktion zum Server zu gesendet [5].

Die Speicherung und Verteilung von Daten und Inhalten sowohl auf der Serverals auch auf der Clientseite benötigt nun zusätzliche Mechanismen, um die Konsistenz der Daten sicher zu stellen. Die Datensicherheit ist ein weiterer Aspekt, dem Beachtung zu schenken ist. Es muss beispielsweise festgelegt werden, ob eine Sandbox für die Speicherung sensibler Daten auf dem Client sicher genug ist.

Ein Modell zur Datenverteilung spezifiziert, wo und wie lange die Daten gespeichert werden. Quelle [LIT03] nennt dazu vier Ebenen, welche die zwei Richtungen beschreiben: Die Verteilung von Daten zwischen Client und Server sowie die Unterscheidung von Daten zwischen dauerhafter oder flüchtiger Speicherung.

- **Flüchtige Daten auf der Clientseite:**
  Daten, die während der Laufzeit zur Verfügung stehen, jedoch mit dem Beenden der Applikation verloren gehen (Laufzeitvariablen)
- **Persistente Daten auf der Clientseite:**
  Daten, die auf der Clientseite gespeichert bleiben, auch wenn der Nutzer die Applikation beendet. RIAs bieten hier, abhängig von der Entwicklungsplattform, Dateien, Datenbanken oder gemeinsam genutzte Objekte (shared objects) als Medium zur Speicherung von Daten an.
- **Flüchtige Daten auf der Serverseite:**
  Daten, die auf der Seite des Servers gespeichert werden, während der Client mit diesem verbunden ist. Die Daten werden wieder gelöscht, wenn die Applikation verlassen wird (beispielsweise Session-Variablen, wie Nutzlogin oder Spracheinstellungen der Applikation).
- **Persistente Daten auf der Serverseite:**
  Daten, die dauerhaft auf dem Server gespeichert werden, unabhängig ob der Client die Anwendung verlässt oder nicht (Dateien oder Datenbanken).

[5] Vgl. [LIT03, S. 2]

Ein typisches Problem bei verteilten Datenbeständen ist die Wahrung der Konsistenz der Informationen zwischen Client und Server. [LIT03] nennt dazu drei verschiedenen Ebenen der granularen Datenkonsistenz (data granularty consistency), die alle Anforderungen von RIAs abdecken:

- **Die field-Ebene:**
  Wird ein einzelner Bereich (field) eines Tupels entweder auf der Client- oder Serverseite verändert, muss die Änderung verbreitet werden, um die entsprechenden Daten auf der Server- bzw. Clientseite zu synchronisieren.

- **Die tupel-Ebene:**
  Wird ein Tupel entweder auf der Client- oder der Serverseite verändert, ist es notwendig, die Änderung zu verbreiten um die Konsistenz der Client- und Server- Daten zu garantieren.

- **Die packet-Ebene:**
  Wird eine Menge von Tupeln (zum Beispiel eine Produktliste) entweder auf dem Client oder auf dem Server verändert, und erfolgt die Synchronisation des Inhalts änderungsgesteuert, so muss die Änderung verbreitet werden, um auch die entsprechenden Daten des Servers bzw. des Clients zu synchronisieren.

[LIT03] schlägt eine mögliche Vorgehensweise vor. Diese umfasst zunächst die Entwicklung des Datenmodellkonzeptes ohne die Beachtung von weiterführenden technischen Aspekten des Entwurfs. Datenmodell und Datenkonsistenz sollen unter Anwendung der bekannten methodischen Ansätze aus der UML- bzw. aus der ER-Modellierung erstellt werden. Diese Modelle müssen den vorhandenen, zuvor bereits aufgenommenen funktionalen und nichtfunktionalen Anforderungen an das System genügen. Im späteren Verlauf der Entwurfsphase müssen dann diese Modelle schrittweise präzisiert bzw. erweitert werden, indem sie beispielsweise auch um die RIA spezifischen Aspekte der Datenspeicherung und Datengranularität angereichert werden. Bei der Wahl von Speichermethoden sind während der Entwurfsphase besonders die Sicherheitsanforderungen an das System zu berücksichtigen: Als einfachste Regel sei hier genannt, dass sensible Daten natürlich nur auf der Serverseite abgespeichert werden dürfen.

3.2.4 Geschäftslogikmodell

In herkömmlichen Web-Anwendungen werden Prozesse nur auf dem Server ausgeführt. Der Client stellt eine Anfrage (request), der Server erzeugt eine neue Seite und sendet diese als Antwort (response) zum Client zurück. Eine RIA unterscheidet sich in der Kommunikationsstruktur von jeder traditionellen Web-Applikation. Sie operiert als Single Page Applikation, in welcher eingebundene Seiten von Client und Server verarbeitet werden können. Passend zur erhöhten Leistungsfähigkeit des Clients, können in Rich Internet Applications sowohl auf dem Server als auch auf dem Client komplexe Operationen ausgeführt werden.

Die Komplexität der Aufgaben, die von RIAs übernommen werden können, erfordern eine reichere Art der verteilten Funktionalitäten im Vergleich zu den traditionellen Web-Anwendungen. Weiterhin werden, sowohl auf Client- als auch auf Serverseite, zusätzliche Mechanismen benötigt, um eine Anforderung (request) abzusenden. Darauf wird im Abschnitt 3.2.6 Kommunikation näher eingegangen.

RIAs erlauben eine Verteilung der Geschäftslogik. Jedoch beschränkt sich die Verteilung nicht nur auf Client- und Serverseite sondern auch auf eine neue Art von Verteilung, die als vermischt (mixed) bezeichnet wird. Clientseitige Geschäftslogik ist dann angemessen, wenn die volle Leistungsfähigkeit des Clients zur Verfügung steht. Serverseitige Geschäftslogik ist wiederum dann angemessen, wenn die Leitungsfähigkeit des Clients beschränkt ist (mobile Endgeräte). Die vermischte Geschäftslogik ist dann notwendig, wenn zwar eine hinreichende Leistungsfähigkeit des Clients verfügbar ist, die clientseitige Logik jedoch Zugriffe auf entsprechende Serveroperationen benötigt. Zum Beispiel wird diese vermischte Variante angewendet, wenn sich der Status der Applikation auf der Seite des Servers verändert hat und die verbundenen Clients benachrichtigt werden müssen. Eine andere Notwendigkeit für die vermischte Variante ergibt sich aus fachlichen Operationen, an die erhöhte Sicherheitsanforderungen bestehen. Diese müssen auf dem Server zur Verfügung gestellt werden.

Laut Quelle [LIT03] ist der konzeptionelle Systementwurf zunächst unabhängig von der Verteilung der Daten. Daher schlägt [LIT03] vor, die Design-Phase in zwei Schritte einzuteilen. Im ersten Schritt soll die Geschäftslogik ohne die Verteilung zwischen Client und Server spezifiziert werden. Danach soll die Spezifikation um den Verteilungsaspekt angereichert werden. In diesem zwei-

ten Schritt soll entschieden und festgelegt werden, wo und wie die einzelnen Arbeitsschritte der Geschäftslogik abgearbeitet werden sollen. [LIT03] weist außerdem darauf hin, dass das Design der Geschäftslogik auf einer deutlich höheren Abstraktionsebene erledigt werden sollte, als der Entwurf des um die Designaspekte angereicherten Modells, bei dem unter anderem auch die Verteilungsapekte berücksichtigt werden müssen. Für das Design der Geschäftslogik und der Geschäftsdaten sollten die UML Diagramme und/oder andere spezifische Modellierungsspachen, wie beispielsweise die ER-Modellierung, verwendet werden. Durch die starke Beziehung zwischen den fachlichen Daten und der auf diesen Daten operierenden Geschäftslogik muss die Verteilung dieser Geschäftslogik zusammen mit der Verteilung der bearbeiteten fachlichen Daten geschehen. Sollen also bestimmte Fachdaten durch den Server verwaltet und verändert werden, so muss auch die Verwaltung der zu den Fachdaten gehörenden Geschäftslogik entsprechend in der Verantwortung des Servers liegen.

3.2.5 Präsentationsmodell

Die Funktionalitäten der Präsentationsschicht ist bei traditionellen Web-Applikationen verhältnismäßig eingeschränkt. So ist es mit Web-Anwendungen nicht möglich, Audio- oder Videoinhalte abzuspielen, wenn nicht zuvor ein entsprechend benötigtes Plugin installiert worden ist. Ebenso unterstützen sie keine reichen Interaktionen wie Drag and Drop oder Animationen. Um solche Interaktionen zu ermöglichen, wäre es notwendig, JavaScript Code zu schreiben, der häufig browserabhängig ist. Die Präsentation konzentriert sich in RIAs auf das Layout, das Definieren von Styles und auf das Verhalten der Applikation in Hinblick auf die Interaktion der Anwendung mit dem Benutzer[6]. Dabei sind die Oberflächen jedoch stark vom Gerät abhängig, welches für diese das Rendering ausführt. Diese unterscheiden sich in Bildschirmauflösung, Unterstützung verschiedener Medien und in den Bedienkonzepten[7]. Daher müssen diese Aspekte bei der Entwicklung von RIA Oberflächen berücksichtigt werden, um eine möglichst hohe Benutzerfreundlichkeit zu erreichen. Durch die Komplexität der Benutzeroberflächen ist häufig eine Einteilung des Präsentationskonzepts innerhalb der RIA Entwicklung je nach Anliegen notwendig. Die folgenden Themen werden dabei von Quelle [LIT03] vorgeschlagen:

- **Räumliche Einrichtung des UI :**
  Diese spezifiziert die Positionierung, Dimension und das Look&Feel der Elemente der Benutzerschnittstelle.

[6] Vgl. [LIT03, S.3]
[7] Beispielsweise ein Touchscreen

- **Zeitliches Verhalten:**
  Dieses definiert die zeitlich-logischen Beziehungen zwischen den Elementen vom User Interface (UI), ohne jedoch dabei die Eingriffe des Nutzers zu betrachten. Dieser Aspekt ist sehr nützlich, um das Verhalten darzustellen, welches temporär auf den Interface-Elementen stattfindet. Dieser Aspekt ist auch nützlich, um Aufrufe der unterliegenden Geschäftslogik zu spezifizieren, welche über zeitgesteuerte Events ausgelöst werden.

- **Interaktives Verhalten:**
  Hier wird die die Interaktion des Nutzers mit dem System spezifiziert, wie z.B. das Verhalten des System nach Mausklick des Benutzers auf ein Element des UI.

Quelle [LIT03] schlägt auch hier wieder vor, das Präsentationsdesign in zwei Schritte aufteilen: Im ersten Schritt soll die Menge an UI-Elementen und deren räumliche Anordnung, Farbe und Größe festlegt werden. Im zweiten Schritt sollen dann die Reaktionen der Elemente auf Ereignisse zusammen mit deren Verhalten und den möglichen Interaktionen festgelegt werden. Natürlich können dabei die Elemente mit mehreren Reaktionen verbunden sein, die gleichzeitig oder zu verschiedenen Zeitpunkten ausgelöst werden können. Das abschließende Design der Oberfläche enthält eine Menge komplexer Verhalten zu jedem Element. Quelle [LIT03] weist darauf hin, dass der Entwickler die Regeln für das Verhalten der interaktiven UI-Steuerelemente sorgfältig festlegen muss, damit die Konsistenz der Präsentationsschicht gewahrt bleibt. Widersprüchliches Verhalten von Komponenten muss natürlich verhindert werden.

Für eine Gruppe von UI-Elementen können mehrere Verhaltensmuster festgelegt werden. Auf diese Weise wird insbesondere die Wiederverwendung der gesamten Gruppe in unterschiedlichen Anwendungen oder Anwendungskomponenten erleichtert. Dieser Sachverhalt soll nun am Beispiel eines virtuellen Einkaufswagens verdeutlicht werden. Im Browser besteht zum Beispiel die Möglichkeit, eine Ware seinem virtuellen Einkaufskorb hinzuzufügen, indem ein Button oder ein Drag and Drop Mechanismus genutzt wird. Auf einem mobilen Endgerät funktioniert im Unterschied dazu der Drag and Drop Mechanismus nicht mehr, stattdessen muss der Button größer präsentiert werden, um diesen auf dem kleineren Touchscreen-Display besser anwählen zu können. Durch die Verwendung der UI-Gruppe kann das Verhalten des gesamten Steuererlements für das Hinzufügen der Ware in den Einkaufswagen dieser gesamten Gruppe zugeordnet werden. Abhängig vom Client kann man nun die Inhalte dieser Gruppe austauschen, ohne die Beziehung des Verhaltens zu den einzel-

nen Steuerelementen anpassen zu müssen, eben weil diese Beziehung an der Gruppe hängt, und nicht an den darin enthaltenen einzelnen Steuerelementen.

3.2.6 Kommunikation

Traditionelle Webanwendungen erlauben in der Regel nur eine synchrone Kommunikation, die von der Seite des Clients veranlasst wird. RIAs dagegen erlauben synchrone und asynchrone Kommunikation zugleich. Die Verteilung der Daten und Funktionen zwischen Client und Server verbessert und vervielfältigt die Funktionalitäten, die in einer Vielzahl von Möglichkeiten ausgelöst durch Ereignisse aufgerufen werden können. Kommunikation ist dabei die übergreifende Angelegenheit bezüglich Datensynchronisation, Verteilung der Geschäftslogik und Präsentation.[8] Im Folgenden soll nun die Kommunikation genauer betrachtet werden.

Kommunikation der Datenschicht bezieht sich hauptsächlich auf das Sicherstellen der Datenkonsistenz und kann dabei von Clientseite (pulling) oder von Serverseite (pushing) ausgelöst werden. Die Granularität der Synchronisation ist dabei von jener der Daten abhängig. Bezüglich des Datenzugriffs beschreibt Quelle [LIT04] die folgenden Methoden aus der Software-Entwicklung[9]:

- **Message-based:**
  Basiert auf einer Nachrichtenfolge, die auf der einen Seite generiert und auf der anderen Seite abgearbeitet wird.
- **State replication:**
  Basiert auf den Logfiles, die die Veränderung aufzeichnen.
- **Replay:**
  Basiert auf Operationen, die auf einer Seite aufgenommen wurden und auf der anderen Seite wieder abgespielt werden.

Üblicherweise werden die Daten auf der Client bzw. auf der Serverseite serialisiert/deserialisiert, um die benötigte Bandbreite besonders für datenintensive RIAs zu verringern.

[8] Vgl. [LIT03, S. 4]
[9] Vgl. [LIT04, S. 50-57]

Kommunikation der Geschäftslogik: Diese befasst sich mit der Kommunikation zwischen verteilen Aufgaben (Tasks) und ist eng mit dem bereits beschriebenen Sachverhalt der vermischten Geschäftslogik verbunden. Client Anwendungen müssen über Ereignisse außerhalb ihrer Ausführungsumgebung benachrichtigt werden, ausgelöst durch Aufrufe anderer Systeme. Dies ist vor allem bei kollaborativen RIAs nützlich, dort finden sowohl Pushing als auch Pulling in synchroner wie asynchroner Variante Anwendung.

Kommunikation der Präsentationsschicht: Diese erfolgt, wie bereits am Anfang des Abschnittes geschrieben, nach dem Single Page Prinzip. Um den ebenfalls angesprochenen Start der Applikation zu verkürzen und die Bandbreitenausnutzung zu verringern, ist es nötig, das Laden der Präsentationsschicht über die Laufzeit der Applikation zu verteilen. Das bedeutet zum Beispiel, dass versteckte Menüpunkte oder Tabansichten für eine erste Ansicht der Applikation zunächst vernachlässigt werden. Dann aber werden diese im Hintergrund asynchron nachgeladen. Dies unterscheidet sich vom On-Demand Laden dahingehend, dass es das Ziel ist, die gesamte Applikation möglichst schnell im Hintergrund nachzuladen. Alternativ besteht jedoch auch die Möglichkeit, die Applikation im Gesamten ohne den Mechanismus des asynchronen Ladens zweitrangiger Elemente zu laden. Dem Nutzer wird dabei ein Ladebalken mit einer Fortschrittsanzeige präsentiert.

Für die zu übertragenden Daten muss bei der Modellierung darauf geachtet werden, dass diese Daten eine Granularität besitzen, die sich in Hinsicht auf die Aspekte der Serialisierung und der Synchronisation als nützlich erweist. Die Wahl dieser Granularität für die zu übertragenden Datenobjekte muss unter anderem von der Menge der während der Laufzeit zu übertragenden Informationen abhängen. [LIT08] ist der Meinung dass die nachrichtenbasierte Variante für Rich Internet Applications viele Vorteile bietet, weil sie gute Fehlertoleranzen und schnelle Antwortzeiten auf asynchrone Ereignisse bietet und weiterhin den Bandbreitenbedarf minimiert. In einer gemischten Geschäftslogik sollten Aufgaben gruppiert werden um die Client/Server Roundtrips zu minimieren. Weiterhin sollen in einer gemischten Geschäftslogik Transaktionsblöcke gebildet werden, damit darauf basierend einfache Datenwiederherstellungen im Fehlerfall realisiert werden können.[10]. Wie bereits im Abschnitt über die Kommunikation innerhalb der Präsentationsschicht angedeutet worden ist, ist es sinnvoll, für das Laden der Applikation Ebenen mit verschiedenen Prio-

[10] Vgl. [LIT05, S.7]

ritäten einzuführen. Auf diese Weise lässt sich die Anwenderfreundlichkeit optimieren, vorausgesetzt dass die Applikation über eine Art Splash-Screen nicht zuvor vollständig geladen wurde.

3.3 Übersicht über spezifische RIA Konzepte

3.3.1 Kategorisierung

Grundsätzlich müssen zwei Arten von Rich Internet Applications unterschieden werden. Diese beiden Arten unterscheiden sich unter dem Aspekt der Laufzeitumgebung (*Runtime Environment*). So läuft die Anwendung im Falle von AJAX innerhalb des Web-Browsers, wogegen andere Technologien ein entsprechendes Browser Plugin benötigen um auf dem Client ausgeführt werden zu können. Genau in diesem Punkt setzen die meisten Kategorisierungen an. Häufig wird bei diesen Kategorisierungen jedoch nicht berücksichtigt, dass der Browser selbst auch aus Laufzeitumgebung dienen kann. So bietet Mozilla mit XUL (*XML User Interface Language*) die Möglichkeit, komplexe User Interfaces innerhalb des Mozilla Firefox' zu erstellen[11]. Wenn man diese Tatsache berücksichtigt, scheint die Kategorisierung nach [LIT05, S.2] am besten geeignet zu sein. Die folgende Tabelle orientiert sich an diesem Vorschlag.

Tabelle 3.1: Kategorische Einteilung aktueller RIA Technologien

Kategorie	**Plattform**	**verbreitete Technologie**
Scriptbasierend	AJAX Libraries	DOJO, ExtJS
Scriptbasierend	AJAX Frameworks	Google Web Toolkit (GWT), Appcelerator
Pluginbasierend	Adobe Flash	Adobe Flex, Laszlo Systems Open Laszlo
Pluginbasierend	Sonstige	Microsoft Silverlight, Curl, Google Gears
Browserbasierend	Mozilla Firefox	XUL
Desktopbasierend	Java	JavaFX, Java Webstart, Canoo UltraLightClient(ULC)
Desktopbasierend	Sonstige	Adobe Integrated Runtime (AIR), Titanium

[11] Siehe `https://developer.mozilla.org/en/XUL`

Die folgenden Abschnitte sollen eine Einführung in die in der Tabelle 3.1 genannten Plattformen bieten. Die konkreten Technologien der Plattformen sollen dagegen ausführlich in Kapitel 4 beschrieben werden.

3.3.2 Adobe Flash

Adobe Flash ist eine Entwicklungsumgebung zur Erstellung multimedialer Inhalte, von Animationen und grafischen Effekten bis hin zu Filmsequenzen, den so genannten Flash Movies. Flash erschien 1997 in seiner von Macromedia entwickelten ersten Version, später wurde Macromedia von Abdobe Systems übernommen, die es heute weiterentwickeln. In traditionellen Webanwendungen wurden häufig Flash-Intros eingesetzt, um zum Inhalt der Webseite hinzuführen. Ein wichtiges Merkmal von Flash ist, dass es sich um ein *streamendes Format* handelt. Von einem Preloader ein Teil der Audio- oder Videodatei vorgeladen und die Datei wird erst dann abgespielt, wenn ein ausreichender Puffer vorhanden ist. Damit ist es möglich, auch bei geringerer Bandbreite eine Audiodatei unterbrechungsfrei anzuhören. Das Streaming der Dateien beruht auf dem Real-Time Messaging Protokoll (RTMP), welches von Adobe Systems entwickelt wurde, um hochperfomante Übertragungen von Audio, Video und Daten zwischen Flash Plattformen zu ermöglichen[12]. Innerhalb dieses Protokolls kommt das RMTP Chunk Stream Protokoll zum Einsatz, welches wiederum eine zeitortierte Auslieferung aller Pakete, auch zwischen multiplen Streams, garantiert.[13] Mit Hilfe dieses Stream-Meachnismus bietet Flash die Möglichkeit, das Single Page Prinzip, eines der zentralen Merkmale einer RIA umzusetzen. Flash Dateien liegen nach der Entwicklung in so genannten SWF-Dateien vor, einem Grafik- und Animationsformat[14]. Um diese Dateien ausführen zu können, ist der *Adobe Flash Player*[15] notwendig, der ebenfalls in einer Lite Version für mobile Geräte zur Verfügung steht.

Flash bietet damit eine technologische Grundlage für Rich Internet Applications. Mit der Einführung von ActionScript in Version 3 ist es zudem möglich geworden, auch Prozesse der Geschäftslogik auf die Seite des Clients zu verlagern und damit den Anforderungen von RIAs zu genügen. Zur Entwicklung von RIAs bietet Flash ein spezielles Framework mit dem Namen *Flex* an, welches im Kapitel 4 näher erläutert werden soll.

[12] Vgl. `http://www.adobe.com/devnet/rtmp/`
[13] Vgl. [URL18]
[14] Vgl. [URL17]
[15] Siehe `http://www.adobe.com/de/products/flashplayer/`

3.3.3 asynchones JavaScript / AJAX

Der Begriff AJAX stammt aus dem Artikel *Ajax: A New Approach to Web Applications*[16] von Jesse James Garrett aus dem Jahr 2005. In diesem Artikel beschreibt er einen neuen Ansatz für die Entwicklung von Web-Applikationen, um sich von den Grenzen der traditionellen Web-Anwendungen zu lösen. Es sei erwähnt, dass Garrett nicht der Erste war, der diesen neuen Ansatz erkannte und beschrieb. Da er aber den Begriff *AJAX* prägte, soll er hier zitiert werden. Garrett erkannte, dass durch die Einfachheit des Webs zwischen den Welten der Webanwendungen und der Desktop-Applikationen in Hinsicht auf die Benutzerfreundlichkeit eine große Lücke klafft.[17]. AJAX stellt seiner Meinung nach eine bedeutende Rolle innerhalb der Möglichkeit des Webs dar. Garrett führt auch die bereits zunehmende Verbreitung vom Web 2.0 zu einem beträchtlichen Teil auf die breite Verwendung von AJAX Technologien zurück. Obwohl AJAX ein häufig verwendeter Begriff ist, bestehen mitunter Unklarheiten, was es genau charakterisiert. AJAX ist keine konkrete Technologie und auch kein Standard oder eine Architektur, sondern vielmehr ein Muster, das auf viele verschiedene Art und Weise implementiert werden kann[18]. In seinem Artikel zählt Garrett die Bestandteile von AJAX auf:

- **Eine auf Standards basierende Präsentation:**
  Die Darstellung der Oberfläche soll wie bei traditionellen Web-Anwendungen durch Standards wie XHTML und CSS umgesetzt werden.

- **Dynamische Anzeige und Interaktion:**
  Mittels DOM (*Document Object Model*) soll der Inhalt der Webseiten dynamisiert und bei Bedarf mit neuen Informationen belegt werden. Die optimale Verwendung von DOM ist dabei für die Dynamik ausschlaggebend.

- **Datenaustausch und Manipulation:**
  Mittels XML und XSLT sollen Daten zwischen Client und Server über XMLHttpRequest (*XHR*) ausgetauscht werden. Wichtig dabei ist, dass die XSD Schemata auf beiden Seiten zur Validierung eingesetzt werden um die Datenkonsistenz zu wahren. Auch wenn der XHR bisher noch kein Standard ist, wird dieser dennoch von allen gängigen Browsern unterstützt[19].

[16] Siehe [URL19]
[17] Vgl. [URL19]
[18] Vgl. [LIT07, S.5]
[19] Siehe http://www.w3.org/TR/XMLHttpRequest/

- **Asynchroner Datenempfang:**
  Die asynchrone Kommunikation zwischen Server und Client durch XMLHttpRequest stellt einen der wichtigsten Teile von AJAX dar. Damit wird es möglich, Bereiche innerhalb der Seite zu aktualisieren, ohne ein vollständiges Neuladen der Seite anzufordern.

- **JavaScript:**
  JavaScript verbindet die einzelnen Teile miteinander. Die XHRs werden über deren API aufgerufen, die Antworten im Anschluss geparsed, validiert und gegebenenfalls auch transformiert, bevor diese durch DOM-Scripting als Ergebnis innerhalb des Browsers dargestellt werden können.

Durch diese Merkmale stellt AJAX eine weitere Möglichkeit dar, RIAs mit ihren spezifischen Merkmalen umzusetzen. Zentrale Rolle spielt dabei die asynchrone Kommunikation, die das Single Page Prinzip ermöglicht.

3.3.4 Java und RIA

Es existieren zahlreiche Ansätze, Rich Internet Applications mit Java umzusetzen. Eine Umsetzung basiert dabei auf so genannten Applets (*Prográmmchen*). Java Applets waren die erste Möglichkeit, das Internet neben den Bereitstellen von Informationen auch als Plattform zum Ausführen von Programmen zu nutzen. Idee des Applets war es, ein universell ausführbares Programm zu schreiben und es in eine Website zu integrieren. Der Client lädt das Programm von der Website und führt es, wenn auch nur innerhalb des Browsers, lokal aus. Als Voraussetzung dafür muss zum Ausführen des Programms eine Java-VM installiert sein. Diese virtuelle Machine (VM) stellt die Laufzeitumgebung des Programms dar. Vorteil der Applets ist, dass unter Beachtung der Sicherheitsregel alle Funktionen der Java SE API zur Verfügung stehen. Der Client verfügt so über eine Möglichkeit, Geschäftslogik auszuführen. Nachteil ist jedoch der große Umfang der Laufzeitumgebung, sowie die Abhängigkeit von der Version der JVM, die benötigt wird um das Applet auszuführen. Weiterhin ist es für Suchmaschinen problematisch, mit Applets umzugehen. Aus diesen genannten Gründen ist die Verbreitung und Beliebtheit der Java Applets seit Ende der 90er Jahre stark gesunken. Für die Entwicklung von Rich Internet Applications stellen demnach Java Applets keine geeignete Möglichkeit dar. Jedoch bietet das Javaumfeld weitere Technologien wie beispielsweise JavaFX oder das Google Web Toolkit (GWT), die im folgenden Kapitel näher erläutert werden.

3.3.5 Mozilla XUL

Bei der XML User Interface Language handelt es sich um eine Beschreibungssprache für grafische Oberflächen. Diese basiert, wie der Name bereits zum Ausdruck bringt, auf XML. Da mit XUL lediglich das Layout der Oberfläche festgelegt und das Design durch CSS definiert wird, ist es möglich, Layout und Design klar voneinander zu trennen. Das hat wiederum den Vorteil, dass beim Austauschen der CSS Datei der Skin der Oberfläche angepasst werden kann, ohne dass jedoch dabei am Layout etwas geändert werden muss. Hinzu kommt, dass die XUL Dateien erst zu Laufzeit kompiliert werden. Dadurch hat der Nutzer die Möglichkeit, während der Laufzeit die Oberfläche nach seinen Wünschen anzupassen. Diese Tatsachen sind für die Benutzerfreundlichkeit sehr förderlich. Problematisch jedoch ist die Tatsache, dass durch das Berechnen der Oberfläche zur Laufzeit sich die Anwendung deutlich langsamer verhält, als sie es mit einer native Oberfläche tun würde. Hinzu kommt, dass das Rendering auf Betriebsystemebene erfolgt. Dadurch ist die Bedienung und Darstellung der Benutzerschnittstelle vom Betriebssystemstandard abhängig. Vor allem der letzte Punkt schadet der Benutzerfreundlichkeit erheblich. Berücksichtigt man weiterhin die Tatsache, das XUL eine reine Technologie für die Darstellung der Oberfläche ist, so erscheint XUL insgesamt für das Umsetzen von Rich Internet Applications eher weniger geeignet.

3.4 Spezialthema RIA und Barrierefreiheit

Der folgende Abschnitt soll das Thema *RIA und Barrierefreiheit* aufgreifen. Betrachtet wird dabei eher die technische Seite des Themas, als der soziale Aspekt. Eine umfangreiche Darstellung der Thematik *Barrierfreiheit* kann im Rahmen dieser Arbeit jedoch nicht geboten werden.

3.4.1 Exkurs: Barrierefreiheit

Als barrierefrei werden in der Regel Systeme bezeichnet, die nicht nur für nicht behinderte Menschen zugänglich sind, sondern auch für Behinderte. Ein mehrstöckiges Gebäude ohne Fahrstuhl ist beispielsweise in diesem Sinne nicht barrierefrei, weil in diesem Gebäude Rollstuhlfahrer keinen Zugang haben zu den Stockwerken oberhalb des Erdgeschosses. Wird das Gebäude jedoch mit Fahrstühlen und Rampen versehen, so wird es zu einem barrierefreien Gebäude. Mit der Vision einer sozialen Gesellschaft ist untrennbar auch die Vision einer barrierefreien Gesellschaft verbunden. Betrachtet man die gesellschaftli-

che Bedeutung des Internets heute und zukünftig und zugleich auch die gesellschaftliche Bedeutung der Barrierefreiheit vor dem Hintergrund einer solchen zukunftsweisenden Vision, so führt das zu einer relativ hohen Gewichtung der Thematik *barrierefreies Intranet/Internet*. Dem barrierefreien oder zumindest *barrierearmen* Intranet/Internet gehört höchstwahrscheinlich die Zukunft.

Nutzer mit Behinderungen benötigen häufig zusätzliche Software, um den im Web veröffentlichten Inhalt für sich angemessen umzuwandeln. In diesem Zusammenhang spielt die Plattform- und Browserunabhängigkeit eine ebenso wichtige Rolle, wie die Möglichkeit, die Website auch auf kleinen Displays in mobilen Endgeräten zu betrachten. Grundlage für diese Geräte- und Softwareunabhängigkeiten sind Standards in der Webprogrammierung.

Brigitte Bornemann-Jeske beschreibt in ihrem Artikel *Barrierefreies Webdesign zwischen Webstandards und universellem Design*, dass es wohlstrukturierte semantische Auszeichnungen sind, durch die Webinhalte in verschiedensten technischen Systemen und für das breite Zielpublikum zur Verfügung gestellt werden können[20]. Sie führt dabei auf, dass Barrierefreiheit am ehesten dadurch erreicht wird, wenn auf der einen Seite klare semantische Strukturierungen und andererseits eindeutige Gestaltungsregeln für individuelle Bedürfnisse berücksichtigt werden. Bornemann-Jeske stellt also einen interessanten Zusammenhang her zwischen der zukunftsweisenden Technologie des semantischen Webs einerseits und der sozialen Möglichkeit eines barrierefreien Webs andererseits. Es ist auch unschwer erkennbar, dass Bornemann-Jeske eine standardisierte Softwareumgebung als grundlegende Notwendigkeit für ein barrierefreies Web hält. User-Agents, wie Screenreadern[21], soll eine Basis geboten werden, den Inhalt einer Website verfügbar zu machen. Dies unterstreicht sie auch mit der Aussage, dass ein valider syntaktisch korrekter HTML Code dem Prinzip der technischen Robustheit entspricht und eine Vorraussetzung dafür ist, dass der Inhalt mit verschiedenen User Agents-Browsern und anderen Anzeigegeräten richtig wieder gegeben werden kann.[22].

[20] Vgl. [URL22, S. 2]
[21] Ein Beispiel ist JAWS: `http://www.freedomsci.de/prod01.htm`
[22] Vgl. [URL22, S. 4]

3.4.2 RIA als Werkzeug zur Umsetzung der Barrierefreiheit

Wie im einleitenden Abschnitt bereits beschrieben, spielt bei der Umsetzung der Barrierefreiheit die Standardisierung der Software eine zentrale Rolle. Damit jedoch Technologien wie Java oder Flash standardisiert werden können, ist es, wie auch Quelle [URL22] in Ihrem Artikel erwähnt, notwendig, grundlegende Verfahren offen zu legen und damit benötigte Schnittstellen zu ermöglichen. Adobe hat diesen Sachverhalt erkannt und 2008[23] die Flash-Spezifikation im *Rahmen des Open-Screens*[24] veröffentlicht. Im Mittelpunkt stand dabei, vorhandene Technik auch für mobile Geräte besser nutzbar zu machen und damit mobile Geräte besser in verteilte Systeme integrieren zu können. Weiterhin ist es für Suchmaschinen dadurch möglich geworden, Flash Dokumente in ihren Suchdurchlauf mit einzubeziehen. RIAs sollen demnach die technischen Voraussetzungen liefern, damit bereits vorhandene Funktionalitäten und Services auch auf mobilen Endgeräten lauffähig sein können. Ebenso notwendig sind offene Standards, damit User Agents den Inhalt korrekt aufbereiten können.

Auf dem Weg zur Umsetzung der Forderung nach einem barrierefreien Intranet/Internet werden RIA Technologien voraussichtlich einen entscheidenden Anteil haben, weil besonders durch RIA Technologien bereits vorhandene Funktionalitäten und Services mit anderen, neuen, mobilen Endgeräten in Verbindung gebracht werden können, die für den barrierefreien Zugang zum Netz sorgen können. Damit dieser Zugang verschafft werden kann, wird RIA wahrscheinlich zusammen mit ARIA zum Einsatz kommen. ARIA bezeichnet *Accessible Rich Internet Application* und ist ein Standard des IBM Mitarbeiters Richard Schwerdtfeder. ARIA soll Lücken in den vorhandenen (X)HTML Spezifikationen füllen um dadurch die Usability für alle Benutzer zu verbessern und Barrieren zu überwinden. (A)RIA Technologien sind auf diese Weise also nicht nur in technologischer Hinsicht zukunftsweisend, sondern leisten auch einen entscheidenden Beitrag in Hinsicht auf eine barrierefreie Gesellschaft.

[23] Siehe `http://www.golem.de/0805/59425.html`
[24] Siehe `http://www.openscreenproject.org/`

3.5 Spezialthema RIA und User Experience

> Most current Web-based applications are ephemeral applications that must be immediately understandable or users will fail. The usability requirements for applications on the Web are much stricter than they ever were for traditional software[25].

Da eine allgemein gültige Definition des Begriffs nicht existiert, sollen an dieser Stelle einige Vorstellungen über User Experience vorgestellt und am Ende zu einem gemeinsamen Ergebnis zusammengefasst werden. Donald Norman prägte als Erster den Begriff *User Experience*, als er bei Apple im Design-Bereich tätig war.[26]. In einem Interview[27] beschreibt er, warum er 1993 den Begriff einführte: „Ich führte den Begriff ein, da ich dachte, dass sich die Begriffe Human Interface und Usability gegenseitig einschränken."Weiterhin beschreibt er im Interview, er wolle mit diesem Begriff alle Aspekte der persönlichen Erfahrung mit dem System, einschließlich des industriellen Design, des Interfaces und der Interaktionen beschreiben.
Etwas allgemeiner beschreibt Jakob Nielsen den Begriff als Umgebung aller Aspekte der Interaktion des Endnutzers, mit einer Firma, deren Leistungen oder Produkten[28]. Da Norman, während er den Begriff einführte, bei Apple tätig war, soll ebenfalls Vorstellung von Apple zum Thema zitiert werden.

> „Die User Experience [...] umfasst das visuelle Aussehen, das interaktive Verhalten und assistierenden Fähigkeiten von Software."[29]

Diese softwaretechnische Betrachtungsweise scheint auch für Rich Internet Applications die am meisten zutreffende Sichtweise zu sein. Die User Experience (*UX*) fasst demnach den *Look*, das Aussehen, den *Feel*, die Interaktion und die *Usability*, die Funktionalität und Intuitivität einer Applikation zusammen. Ziel einer RIA muss es sein, eine deutlich höhere UX zu erreichen, als es eine traditionelle Web-Anwendung schafft. Grundlagen für eine erhöhte UX stellen dabei das Single-Page-Prinzip, sowie reiche Interaktionsmöglichkeiten wie Drag and Drop dar. Allein diese technische Umsetzung reicht jedoch nicht aus. Die Applikation muss dem Nutzer auch verständlich vermitteln, wie er diese Möglichkeiten der Interaktion für sich nutzen kann. Diese selbsterklärende Vermittlung muss dabei, laut dem Zitat zu Beginn, am besten auf dem *ersten Blick* geschehen.

[25] Vgl. [URL24, S. 5]
[26] Vgl. [URL25]
[27] Vgl. [URL26]
[28] Vgl. `http://www.nngroup.com/about/userexperience.html`
[29] Vgl. `http://developer.apple.com/ue/`

4

Rich Internet Application - spezifische Technologien

Mit Hilfe der gewonnenen Erkenntnisse über die grundlegenden Merkmale und Konzepte von RIAs, soll in diesem Kapitel eine technische Betrachtung konkreter Technologien, mit deren Hilfe die spezifischen Merkmale von RIAs umgesetzt werden können, im Mittelpunkt stehen.

4.1 AJAX Spezifikation

Bereits im vorherigen Kapitel wurde der Ursprung des Begriffs und die Merkmale von AJAX erläutert. Ziel dieses Abschnittes ist es die konkrete, technische Umsetzung aufzuzeigen.

Im AJAX - Modell läd der Browser zunächst eine Initialseite bestehend aus HTML, CSS und Java Script. Führt der Nutzer eine Aktion aus, etwa einen Klick auf einen Button, wird kein Request vom Browser an den Server gesendet, sondern stattdessen eine JavaScript Funktion aufgerufen. Diese Funktion ist als ein EventListener registriert und sendet mit Hilfe eines XMLHttpRequests einen HttpRequest an den Server. Dies geschieht im Hintergrund und nicht synchron mit den Eingaben des Nutzers, so dass dieser weitere Daten eingeben kann, während bereits versteckt mit dem Server kommuniziert wird. Sendet der Server seine Antwort (Response) an den Browser zurück, wird die-

se von einer weiteren Funktion (Callback-Funktion) entgegengenommen. Mit Hilfe der DOM API kann diese Funktion die aktuelle Seite modifizieren, indem sie innerhalb des DOM Baumes, Zweige hinzufügt, verändert oder löscht. Das Ergebnis ist für den Nutzer unmittelbar sichtbar und auf ein komplettes Neuladen der Seite kann verzichtet werden kann. Auch kann der Nutzer nicht unterscheiden ob nur eine clientseitige Veränderung der Seite oder ob eine Kommunikation mit dem Server stattgefunden hat.

Die beiden Grafiken verdeutlichen noch einmal die traditionelle und Kommunikation nach dem AJAX-Prinzip. In beiden Fällen sollen auf einer Seite zwei Komponenten (grün) durch zwei neue Komponenten (rot) ersetzt werden. Ob es sich hierbei um Bilder, Videos oder nur Text handelt ist unerheblich.

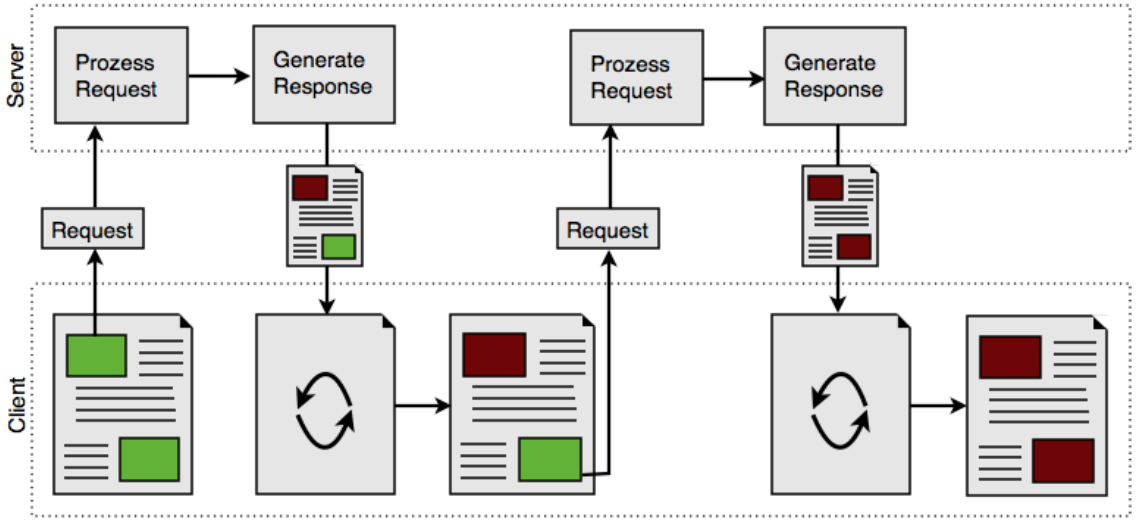

Abbildung 4.1: Request-Respone-Ablauf einer herkömmlichen Webanwendung in Anlehnung an [LIT09]

Abbildung 4.1 illustriert den Request-Response-Ablauf einer traditionellen Webanwendung, welcher an dieser Stelle kurz beschrieben werden soll:

- Durch den Klick auf die Komponente (grün), löst der Nutzer ein Event aus, dass eine Anfrage an den Server sendet.
- Diese Anfrage wird von Server entgegen genommen und im Anschluss eine Antwort generiert.
- Die Antwort in Form einer kompletten, *neuen* Seite wird an den Client zurückgesendet.

- Darauf hin lädt der Browser die vollständige Seite neu.
- Nach dem Neuladevorgang ist die neue Komponente (rot) sichtbar.

Der Ablauf für die zweite Komponente erfolgt analog, so dass bei der Veränderung der beider Komponenten die komplette Seite zweimal neu geladen werden muss. Abbildung 4.2 stellt dagegen den asynchronen Ablauf einer AJAX - unterstützten Anwendung dar.

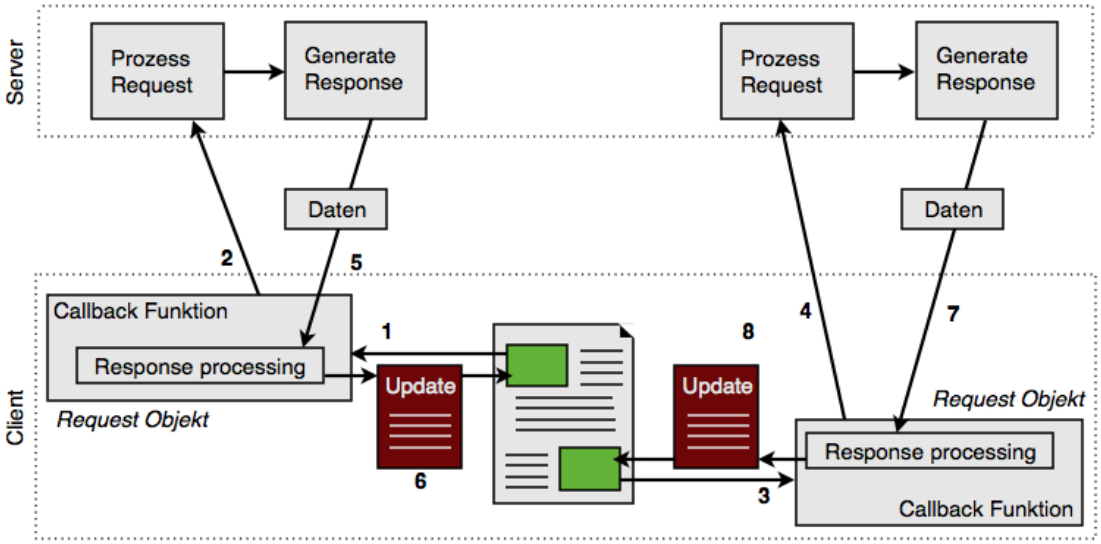

Abbildung 4.2: Asynchroner Ablauf einer AJAX Anwendung in Anlehung an [LIT09]

1. Der Nutzer löst Event aus, welches ein XHR Objekt wird erzeugt. Zudem wird dem XHR Objekt die Callback-Funktion übergeben.
2. Das XHR Objekt sendet die Anfrage an den Server.
3. Der Nutzer löst ein weiteres Event aus, worauf ein XHR Objekt erzeugt und diesem die Callback-Funktion übergeben wird.
4. Das zweite XHR Objekt sendet die Anfrage an den Server.
5. Der Server sendet seine Antwort in Form einer XML Datei, die ausschließlich Daten enthält. Die Callback-Funktion nimm diese entgegen und prüft den Status der Antwort.
6. Beim Status *complete* werden die Daten aus der XML Datei ausgelesen und in den DOM-Baum übertragen. Der veränderte Zweig im DOM-Baum wird anschließend neu geladen.

Die Schritte 7 und 8 erfolgen analog zu den Beschriebenen(5 und 6), der ersten Komponente. Anzumerken ist, dass die oben beschriebene Reihenfolge nicht zwingend vorgegeben ist. Denkbar wäre es auch, dass die erste Komponente (1.) erst mit den Schritten 7 und 8 neu geladen würde. Tatsächlich spielt dies auch keinerlei Rolle, da es nur wichtig ist das nach den 8 Schritten beide Komponenten verändert wurden.

4.1.1 AJAX Frameworks

Es existiert eine Vielzahl an Frameworks, welche die Funktionalitäten von AJAX umsetzten. Jedoch unterscheiden sich diese im Umfang und Umsetzung der Funktionalitäten. Die Frameworks lassen sich in drei Kategorien einteilen, die nun folgend mit einem Beispiel kurz vorgestellt werden.

4.1.2 Addon Frameworks

Ziel von Addon Frameworks ist es, die Kommunikation zwischen Server und Client zu einem Framework zusammenzufassen. Dabei werden die AJAX-Funktionalitäten zu bereits bestehende Server-Frameworks oder Technologien hinzugefügt. Vorteil dieser Frameworks ist, dass diese optimal für die eine spezielle Technologie umgesetzt wurden. Beispiele für solche Addon Frameworks sind AJAX JSP Tags[1] und AJAX JSF Frameworks wie IceFaces[2] oder MyFaces[3]

Die AJAX JSP Tag Bibliothek ist eine Sammlung von JSP Tags, die die Benutzung von AJAX Technologien innerhalb von JavaServer Pages vereinfachen. Mit Hilfe dieser JSP Tags ist es möglich die Funktionalitäten die in JavaScript geschrieben werden müssen, ohne tiefe Kenntnisse von clientseitiger Programmierung einfach hinzuzufügen[4]. Die Tag-Library bietet Unterstützung für live-Updates in Komponenten wie Autocomplete, basierend auf dem Buchstaben der in ein Textfeld eingegeben wurde; balloon-Popups, zur Hervorhebung von Inhalt oder die Möglichkeit einzelnen Formularfelder neu zu laden. Die Implementierung der Bibliothek basiert dabei auf einer Kombination von Java Klassen und JavaScript Dateien und ist damit weitgehend Betriebssystem unabhängig. Zum Ausführen ist jedoch mindestens ein Firefox der Version 2.0 oder der Internet Explorer 6.0 notwendig.

[1] Siehe `http://ajaxtags.sourceforge.net/`
[2] Siehe `http://www.icefaces.org/main/home/`
[3] Siehe `http://myfaces.apache.org/`
[4] Vgl. `http://ajaxtags.sourceforge.net/`

4.1.3 Integrated Component Frameworks

Das Ziel von komponentenbasierten Frameworks ist es AJAX vollständig zu integrieren und die clientseitigen Funktionalitäten in einer Hochsprache wie z. B. Java zu implementieren. Die Entwicklung und die Programmsyntax gleichen dabei fast der eines Java-Swing Clients. Ein weiteres Merkmal dieser Frameworks ist es, die Entwickler von Browser-Inkompatibilitäten von JavaScript im Zusammenspiel mit CSS und HTML möglichst fernzuhalten. Zudem bieten diese Frameworks meist Entwicklungsumgebungen mit Test-und Debuggingmöglichkeiten. Ein bedeutendes Beispiel für komponentenbasierte Frameworks ist das Google Web Toolkit (GWT)[5].

Das GWT ermöglicht es vollständige Anwendungen, Client und Server eingeschlossen, in Java zu verfassen. Dabei generiert es den gesamten JavaScript Code auf Basis des Java-Codes[6]. Das Google Web Toolkit bietet zudem eine Möglichkeit sowohl Client als auch Server zu Debuggen. Ermöglicht wird dies durch die GWT-Clientbibliotheken, welche in Java geschrieben sind und erst dann zu JavaScript und HTML konvertiert werden, wenn der Client kompiliert wird. In einem *Hosted Mode* bietet das GWT die Möglichkeit den Java Code zu debuggen und gleichzeitig im Browser ablaufen zu lassen. Diese Tatsache funktioniert dadurch, da die GWT-Shell zum einen dafür sorgt, dass die Dateien über ihren später durch das Kompilieren festgelegten Pfad verfügbar sind und zum anderen, dass die Java Dateien dynamisch ausgeführt, und später im Browser als JavaScript abgearbeitet werden können[7].

4.1.4 Java Script Frameworks

Bei dieser Kategorie von Frameworks handelt es sich um reine JavaScript Bibliotheken, die nicht auf ein bestimmtes Web Framework festgelegt sind. Damit ist dieser Ansatz besonders flexibel. Browser-Inkompatibilitäten werden durch die Cross-Browser Lösungen der einzelnen Bibliotheken weitgehend ausgeschlossen. Vorteil dieser Art Frameworks ist es, dass sich die API einfach in den HTML Code einbinden lässt und meist umfangreiche Funktionalitäten bietet. Dabei werden aber im Gegensatz zu den Addon-Frameworks keine Anforderungen an die Servseite gestellt. Ein Beispiel für diese Framework- Kategorie ist das DOJO JavaScript Toolkit[8].

[5] Siehe `http://code.google.com/intl/de/webtoolkit/overview.html`
[6] Vgl. [LIT11, S. 196]
[7] Vgl. [LIT12, S. 21]
[8] Siehe `http://www.dojotoolkit.org/`

DOJO ist eine modulare JavaScript-Bibliothek und beinhaltet eine Anzahl von Widgets (vorgefertigen Bausteinen) wie Menüs, Tab, Tabellen, Listen, Online-Editoren für formatierten Text. Für die asynchrone Kommunikation stellt DOJO einen Wrapper (dojo.io.bind) zur Verfügung. Dieser Wrapper unterstützt verschiedene Transportmechanismen(weitere neben dem XHR) und Datenformate. Der Aufbau der Bibliothek ist modular, das heißt die einzelnen Funktionalitäten sind in Ordnern und Unterordner geschachtelt. Ein Bootstrap-Skript initialisiert einen Wurzelnamensraum *dojo* und eine Reihe von Paketnamensräumen wie *io* und *event*, zudem besteht die Möglichkeit der Einbindung neuer Namensräume, um eigene Zusatzpackete einzubinden[9]. Als weiteres Feature besteht die Möglichkeit Daten auf der Clientseite, mit DojoStorage, zu speichern. Dazu wird jedoch die Einwilligung des Benutzers benötigt. Wie genau die Speicherung der Daten abläuft hängt dabei vom jeweiligen Browser ab[10].

AJAX bietet für die Entwicklung von Rich Internet Applications zahlreiche Ansätze. Welcher Ansatz zu wählen ist, hängt dabei vom Umfeld der jeweiligen Applikation ab. Addon Frameworks bieten in einer bereits bestehenden Anwendung die Möglichkeit, Funktionen so anzureichen, das Teile dem Single Page Prinzip entsprechen. Um jedoch eine RIA mit einer Desktop-ähnlichen UX zu entwickeln, ist der Funktionsumfang solcher Framework zu gering. Dafür eignen sind die anderen beiden Arten von Frameworks besser. Während bei einer Neuentwicklung Toolkits wie das GWT zu bevorzugen sind, ist bei der Erweiterung einer bestehen Web-Applikation die Nutzung eines JavaScript Toolkits sicher vorzuziehen, da bei diesen keine serverseitigen Änderungen notwendig sind. Zusammenfassend ist zu sagen, dass AJAX eine gute Grundlage für die Entwicklung von Rich Internet Applications bietet. Der große Vorteil der Technologie ist, dass für die Ausführung kein Plugin notwendig ist[11].

[9] Vgl. [URL27]
[10] Vgl. [URL28]
[11] Der Browser muss jedoch JavaScript unterstützen

4.2 Adobe Flex

Adobe Flex[12] ist das auf Flash basierende Entwicklungsframework von Adobe. Für die Entwicklung von RIAs bietet Flex die Sprachen MXML und Actionscript. Das deklarative MXML dient dazu die Views der Applikationen zu stellen, wogegen das Actionscript, als objektorientierte Sprache verwendet werden kann, um die clientseitige Geschäftslogik, den Controller und die Eventbehandlung zu implementieren. Dabei ist MXML ein Abtraktionslayer über Actionscript. Beim Kompilieren wird MXML in Actionscript überführt und letzteres schließlich zur im Flashplayer ausführbaren SWF-Datei. Folgendes Codebeispiel soll die Verwendung von MXML illustrieren.

```
<? Xml version=‘‘1.0‘‘ encoding=‘‘UTF-8‘‘?>
<mx:Application xmlns:mx=’’http://www.adobe.com/2006/mxml’’
   layout=’’absolute’’>
      <mx:Label id=’’myLabel’’ text=’’Hello World !’’ />
</mx:Application>
```

Zu erkennen ist, dass es sich um eine typische XML-Sprache handelt, die korrekt geschachtelt und wohlgeformt sein muss. Namespace der MXML Tags, die Teil des Flex-Frameworks sind, werden in der Regel mit *mx* bezeichnet. Um eigene Komponenten definieren zu können, werden dazu weitere Namespaces definiert. <mx:Application> legt als Rootelement den Kontext für die gesamte Anwendung fest. Das layout-Attribut kann dabei zwischen drei Werten variieren. Zur Verfügung stehen neben dem im Beispiel verwendeten *absolute* auch *vertical* und *horizontal*. Während bei absolut die Kindkomponenten explizit mit x-und y-Koordinaten versehen werden müssen, implementieren die letzteren einen einfachen Layout-Manager. MXML bietet damit die Möglichkeit schnell und übersichtlich grafische Oberflächen zu entwickeln.

Actionscript wurde von Macromedia erstmal mit Flash 4.0 bereitgestellt und sollte zu dieser Zeit für die Steuerung von Fashmovies dienen. In der aktuellen Version 3 ist Actionscript jedoch zu einer umfangreichen Programmiersprache geworden die die Entwicklung hochkomplexer Anwendungen mit großen Datenmengen ermöglicht[13]. Folgender Codeausschnitt soll die Verwendung Syntax von Actionscript verdeutlichen:

[12] Siehe `http://www.adobe.com/de/products/flex/`
[13] Vgl. [LIT13, S. 28]

```
package
{
   public class Hello
   {
      /* Kommentar */
      public function sayHello(name:String = ,,‘‘): String
      {
         var helloString:String;
         helloString= ‘‘Hello’’ +name;
         return helloString;
      }
   }
}
```

An diesem einfachen Beispiel erkennt man die Ähnlichkeiten zu Java oder C#. Dieser Code wird, um diesen auszuführen, von einem Compiler in Bytecode umgewandelt. Dieser Bytecode wiederum wird im Rahmen des Kompilierungsprozesses in die finale SWF geschrieben. Diese kann dann vom Flash Player ausgeführt bzw. interpretiert werden.

Für die Anforderungen an die Kommunikationen bietet Abobe Flex verschiedene Möglichkeiten von einfachen XML, über das Action Message Format (AMF) bis zum Real Time Messaging Protokoll (RTMP)[14]. Somit sind möglich Merkmale wie das Single Page Pinzip problemlos umzusetzen. Auf Grund der einfachen Erstellung von Oberflächen mit MXML und den breiten Funktionen von Actionscript clientseitige Geschäftslogik zu implementieren, stellt Flex eine sehr gute Plattform für die Entwicklung von RIAs dar. Einzig die Notwendigkeit des Flash Players zum Ausführen der Programme stellt einen Nachteil dar.

4.3 OpenLaszlo

Mit OpenLaszlo[15] bietet Laszlo Systems eine weitere Möglichkeit für die Entwicklung von Rich Internet Applikationen. Das Besondere an OpenLaszlo ist, dass die Entwicklung der Anwendung auf der Seite des Servers erfolgt. Im Anschluss ist es dann möglich die Quelldateien, zu DHTML oder zu SWF umzuwandeln und so auf der Seite des Client auszuführen. OpenLaslo bezeichnet

[14] Vgl. [LIT13, S. 6]
[15] Siehe `http://www.openlaszlo.org/`

sich dadurch als einzige überall-lauffähige, nicht eingeschlossene Internetplattform[16]. Eine Laszlo Applikation besteht aus LZX Dateien, welche eine Mischung aus XML und Script Dateien darstellen. Diese Dateien werden dann mit Java in die erwähnten Formate umgewandelt. Zur Erstellung von grafischen Oberflächen bietet die Plattform eine Reihe von Standardkomponenten an, die im Aussehen (*Skin*) verändert werden können. Ein einfaches Fenster kann mit OpenLaszlo in der folgenden Weise erzeugt werden.

```
<canvas width="250">
  <window x="10" y="10" width="200" height="200" >
      <text>Hello World!</text>
      <button>Hello!</button>
   </window>
 </canvas>
```

Dieser Quellcode erzeugt auf einen 250 Pixel breiten Canvas ein 200 x 200 Pixel großes Fenster an der Position 10, 10. Im Fenster befindet sich der Text *Hello World!* und eine Button mit der Beschriftung *Hello!*. Dieser Quellcode würde nun in einer LZX Datei abgespeichert und im Anschluss wahlweise in DHTML oder Flash umgewandelt werden.

OpenLaszlo bietet mit der Möglichkeit die Darstellungstechnologie des Presentationlayers auszuwählen, den Vorteil herstellerunabhänger RIAs entwickeln zu können und dadurch einer größere Browserunabhänigkeit zu erreichen. Ein nicht zu unterschätzender Nachteil sind jedoch die Performance-Einbußen der Anwendungen, die auf Grund der Konvertierung in die entsprechende Technologie entstehen. Daher existieren zahlreiche Tipps und Guides zur Leistungssteigerung, vor allem beim Startvorgang[17].

4.4 JavaFX Frameworks

JavaFX[18] ist die Plattform von Sun Microsystems zur Entwicklung von Rich Internet Application. Hinter dem Begriff JavaFX verbergen sich die beiden Technologien JavaFX Script und JavaFX Mobile. JavaFX Script stellt dabei als zetraler Kern der gesamten Technologie die Mittel zur Verfügung, um allgemeine, visuelle und hochleistungsfähige Anwendungen auf der Basis von Java

[16] Orig. *The only run-anywhere, no-lock-in internet plattform.*
[17] Vgl. [URL29] und [URL30]
[18] Siehe `http://javafx.com/`

zu erstellen[19]. JavaFX Mobile ist das Softwaresystem, das für mobile Endgeräte über eine OEM-Lizenz zur Verfügung steht. Da Java Mobile Applikationen ebenfalls mit Hilfe vom JavaFX Script entwickelt werden, soll im Folgenden der Schwerpunkt auf die Skriptsprache JavaFX Script gelegt.

Im Vordergrund dieser Sprache steht die intuive und effiziente Gestaltung grafischer Oberflächen und die Anreicherung mit visuellen Effekten. Daher handelt es sich bei JavaFX um eine Domain Specific Language, also einer Sprache die nur für ein bestimmtes Problemfeld entworfen wurde. JavaFX zeichnet sich durch die Objekt Literal Syntax aus, bei der JavaFX Objekte die Basiselemente darstellen und deren Attribute zur Definition des Objektstatus dienen. Folgendes Beispiel soll die Syntax verdeutlichen[20]:

```
Adresse {
    strasse: „Breiter Weg 12“;
    plz: „33547“;
    stadt: „Neustadt“;
}
```

Adresse ist das Objekt, das durch seine Attribute strasse, plz und stadt, mit deren Werten, deklariert wird. Dieses deklarative Konzept wurde konsequent in JavaFX umgesetzt. An einem weiteren Beispiel soll der Aufbau eine einfachen grafischen Oberfläche mit JavaFX aufgezeit werden.

```
Stage {
  title: „Hello World !“
    scene: Scene {
      content: [
        Text {
          x: 10 , y: 10
          content: “Hello World !”
}]
}
}
```

Dieser Ausschnitt stellt den vollständigen Code zum Erzeugen eines einfachen Frames, mit dem Titel und einem Textfeld *Hello World !*. Wie zu sehen wird, die gesamte UI wird als Baumstruktur deklariert, wobei *Stage* den Rahmen

[19] Vgl. [LIT14, S.15]
[20] Angelehnt an `http://java.sun.com/javafx/1/tutorials/core/usingObjects/`

bildet und eine Scene als Container für die restlichen visuellen Komponenten dient. Auffällig hierbei ist der Bezug zum SzeneGraph-Prinzip aus der Grafikprogrammierung. Die Aussage, die zentrale Metapher zur Spezifizierung von Grafiken und Nutzerinteraktionen in JavaFX sei ein Szenegraph[21], bekräftig dies nur all zu sehr. JavaFX kann bei der Oberflächengestaltung auf eine Reihe von visueller Komponenten, Effekte und Utilities zurückgreifen. In der Regel handelt es sich dabei um Komponenten von Wrapper-Klassen der Java-Eigenen Swing und 2D API, welche die Verwendung deutlich vereinfachen. Außer Swing und Java-2D Komponenten bietet JavaFX Möglichkeiten zum Audio- und Video-Support, synchoner und asynchroner HTTP-Request-Unterstützung, sowie einem Pull Parser zur einfachen Verarbeitung vom XML und JSON Dokumenten.

JavaFX stellt mit seinem Funktionsumfang eine weitere Möglichkeit dar Rich Internet Applications zu entwickeln, wobei Quelle [LIT14] bemerkt, dass JavaFX nicht primär dafür vorgesehen ist innerhalb eines Browsers zu laufen. Im Gegenteil, JavaFX versucht sogar die Fesseln des Browsers zu sprengen[22]. Somit stellt JavaFX eher eine Möglichkeit dar RIAs zu entwickeln, welche direkt auf dem Desktop oder auf mobilen Geräten laufen. Wie bereits am Aufbau des Programmstruktur festgestellt liegt der Schwerpunkt dabei auf der schnellen Entwicklung modernen Oberfächen. Daher ist bei der Entwicklung von RIAs mit vielen multimedialen Inhalten und grafischen Elementen JavaFX möglicher Weiße besonders geeignet.

4.5 Silverlight

Die Version 1.0 von Silverlight[23] wurde am 5. September 2007 von Mircosoft veröffentlich. Sie stellte ein einfaches Präsentationsframework, das mit XAML (*Exensible Application Markup Language*) arbeitet. XAML ist eine XML-basierte Markup Sprache die von Microsoft für das Windows Presentation Framework (WPF) entwickelt wurde. Die derzeitig aktuelle Version 3 ist für Windows und Mac OS X und für die Browser Internet Explorer, Firefox und Safari verfügbar. Bereits mit Version 2 war es mit Silverlight möglich, Anwendungen mit jeder beliebigen .Net Sprache zu entwickeln. Das Programmiermodell basiert auf WPF, in dem jedoch nur ein Teil der Funktionen in

[21] Vgl. [LIT15, S. 15]
[22] Vgl. [LIT14, S. 21]
[23] Siehe `http://silverlight.net/`

Silverlight zur Verfügung steht. Gründe dafür sind die plattformübergreifende Unterstützung und die Optimierung der Dateigröße der Laufzeitumgebung (Runtime). Die fehlende Hardwareunterstützung wurde mit Version 3 hinzugefügt[24]. Damit stellt Silverlight, laut [LIT16], einen ausgewogenen Kompromiss zwischen Kompaktheit und Komplexität dar.

Seit Version 2 entspricht die Common Language Runtime (CoreCLI) von Silverlight, der des .Net 3.0 Frameworks und ist damit in der Lage alle .Net Sprachen auszuführen. Dadurch lassen sich XAML Dateien, die das Layout der Anwendung definieren, durch .Net Code in Funktion und Interaktivität erweitern. Silverlight bietet damit flexible Möglichkeiten clientseitige Geschäftslogik umzusetzen. Zur RIA-typischen Datenverarbeitung ist es durch die API möglich Daten zu serialisieren und externe Datenformate wie XML, RSS und JSON zu bearbeiten. Da Silverlight ebenfalls die Funktionen von LINQ[25] unterstützt, ist es problemlos möglich Datenquellen wie Datenbanken oder auch XML-Dateien abzufragen. Zur persistenten Speicherung von Daten auf der Seite des Clients bietet Silverlight so genannten *isolate local storage* oder auch *isostorage*. Mit Hilfe dieses Speichers können Dateien bist zu 1MB pro Url gespeichert werden. Um eine Vergrößerung des Speichers zu erlauben, ist jedoch die Zustimmung des Nutzers vonnöten.

Zur Kommunikation wartet Silverlight mit einer umfangreichen Netzwerkunterstützung auf. Neben Services wie WCF, REST oder RSS bietet es auch HTTP Dienste, die dank Threading-Biblotheken auch die asynchrone Kommunikation unterstützen und damit das Single Page Prinzip, als ein wichtiges Merkmal von Rich Internet Applications, möglich macht. Auch für die Barrierefreiheit von RIAs, bietet Silverlight Unterstützung. Durch serverbasierte Business Objekte in Kombination mit ASP.NET Controls und Site Maps lassen sich datenbankbasierte Inhalte automatisch in HTML widerspiegeln, welches problemlos von Suchmaschinen indiziert werden kann[26]. Durch diese Möglichkeit, Silverlight SEO (Search Engine Opmization) genannt, ist die entwickelte RIA jedoch auch für Screenreader besser lesbar.

[24] Siehe `http://www.microsoft.com/germany/net/silverlight/newfeatures.aspx`
[25] Siehe `http://msdn.microsoft.com/de-de/library/aa479865(en-us).aspx`
[26] Siehe `http://www.silverlightshow.net/items/Silverlight-SEO.aspx`

4.6 Zusammenfassung

Dieser Abschnitt soll die in dieser Arbeit vorgestellten Technologien zur Entwicklung von Rich Internet Applications noch einmal übersichtlich zusammenfassen. Dazu ist zu jeden Framework bzw. zu jedem Toolkit die Plattform, ebenso wie die dazugehörige Quelle im Internet angegeben. Hinzukommen die beschrieben Merkmale der Technologie. Anzumerken ist, das sich die Browserunabhänigkeit im Falle von AJAX eine Unterstützung von JavaScript voraussetzt.

Framework/ Toolkit	**Plattform**	**URL**
AJAX JSP Tags	AJAX	`http://ajaxtags.sourceforge.net/`
- Erweiterung des Funktionsumfangs des bisherigen Frameworks - gleiche verwendetet Sprache - kein Browserplugin notwendig - mindestes Firefox 2.0 oder Internet Explorer 6.0		

Google Web Toolkit (GWT)	AJAX / Java	`http://code.google.com/intl/de/webtoolkit/overview.html`
- Client- und Server-Code vollständig in Java implementierbar - IDE mit Debugging-Funktion - kein Browserplugin notwendig - browser- und plattformunabhänig		

Dojo JavaScript Toolkit	AJAX / JavaScript	`http://www.dojotoolkit.org/`
- JavaScript Bibliothek mit zahlreichen vorgefertigten Komponenten - reine clientseitige Erweiterung, keine Änderung der Serverseite - kein Browserplugin notwendig - browser- und plattformunabhänig		

Adobe Flex	Flash	http://www.adobe.com/de/products/flex/
- Entwicklungsframework auf Flashbasis - umfangreiche IDE für die Entwicklung - FlashPlayer als Plugin für den Browser notwendig - FlashPlayer verfügbar für Windows, Mac OS X und Linux		

OpenLaszlo	DHTML / Flash	http://www.openlaszlo.org/
- browser- und plattformunabhänig durch Konvertierung in DHTML/Flash - vorgefertigte Standardkomponenten - Browserplugin zum Teil notwendig - Performanceoptimierungen seitens des Entwicklers notwendig		

JavaFX	Java / JavaFX Script	http://javafx.com/
- basiert auf Java / JavaFX Script - JavaFX mobile als Client für mobile Geräte - grafik-orientierte Entwicklung von RIAs - JRE als Browserplugin notwendig - plattform- und geräteunabhänig		

Silverlight	.NET	http://silverlight.net/
- Entwicklung von RIAs auf .Net Basis in allen .Net Sprachen - umfangreiche Unterstützung aller RIA-spezifischen Merkmale - Browserplugin notwendig - offiziell nur für Windows und Mac OS X		

5 Auswirkungen von RIAs auf die Entwicklung

Mit Hilfe der gewonnenen Erkenntnisse aus den vorhergehenden Kapiteln, soll in diesem Kapitel eine ausgiebige Erörterung der Auswirkungen auf die Entwicklung von Rich Internet Application folgen. Dazu sollen Thesen aufgestellt und an diesen die Entwicklungsauswirkungen beleuchtet werden. Mit einem Beispiel sollen die Thesen unterstützt werden.

5.1 Auswirkung auf die Entwicklung

5.1.1 Steigende Ansprüche an die Software Ergonomie

> *These* α: Die Anforderungen an das Verhalten und die Ergonomie von Web-Anwendungen im Intranet und im Internet werden in Zukunft steigen.

Mit weiterhin zunehmendem Konkurrenzdruck ist von einem starken Wettbewerb von Anbietern im Internet um Kunden auszugehen. Daher werden die dort agierenden Wettbewerber unter anderem versuchen, sich bei ihrem Wettbewerb um Kunden mit entsprechenden Verbesserungen auch bei der Software-Ergonomie gegenseitig zu überbieten. Gemeint sind hier zum Beispiel Suchmaschinen, B2C (Business to Consumer) Portale oder B2B (Business to Business) Marktplätze. Was das Intranet angeht, so ist bei weiter-

hin zunehmendem Kostendruck davon auszugehen, dass eine deutlich verbesserte Software-Ergonomie bei häufig verwendeten Intranet-Applikationen zur Ersparnis von Arbeitszeit und somit von Geld beitragen wird. Gemeint sind hier zum Beispiel Zeiterfassungssysteme, Mitarbeiter-Einsatzplanungssysteme, Kalender oder Mail-Clients. Diese Entwicklungen spielen sich vor dem Hintergrund von immer mehr und immer billiger zur Verfügung stehenden Rechenleistungen und Speicherkapazitäten, ab. Deutliche Verbesserungen der Ergonomie von Webanwendungen in Internet und Intranet werden also durch die Verbesserungen der zugrunde liegenden Rechenleistungen und Speicherkapazitäten ermöglicht. All diese eben genannten Entwicklungen sprechen dafür, dass die Anforderungen an das Verhalten und die Ergonomie von Web-Anwendungen im Intranet und Internet in Zukunft zunehmen werden.

5.1.2 Verstärkte Bedeutung der Präsentationsschicht

> *These* β: Technologien für Webanwendungen, die sich dem Presentation Layer zuordnen lassen, werden in Zukunft an Bedeutung gewinnen. Im Zusammenhang damit, werden auch RIA Technologien in Zukunft an Bedeutung gewinnen.

Um diese gestiegenen Anforderungen an die Ergonomie von Web-Anwendungen in Zukunft mit vertretbaren Preisen erfüllen zu können, werden entsprechend weiter entwickelte, mächtigere Technologien für Webabwendungen benötigt, die sich dem Presentation Layer zuordnen lassen. Daher ist davon auszugehen, dass diese Technologien in Zukunft an Bedeutung gewinnen werden. RIA Technologien gehören zu diesem Typus von Technologien und werden daher voraussichtlich ebenfalls an Bedeutung gewinnen. Zu den RIA Technologien gehören die in dieser Arbeit vorgestellten Technologien wie beispielsweise Adobe Flex, AJAX Technologien und Java FX.

5.1.3 Konsolidierung von RIA Technolgien/Frameworks

> *These* γ: Die aktuell vorhandenen RIA Frameworks, sowie deren Integrationen in die vorhandenen Technologien sind sehr komplex. Es ist von einer weiteren Konsolidierung der derzeit vorhandenen Frameworks am Markt auszugehen. Dabei werden sich diejenigen Frameworks durchsetzen, die Benutzerkomfort, sowie schnelle Entwicklungszeiten bieten und mit angemessenem Aufwand erlernbar sind.

Damit sich RIA Technologien für eine breite Kommerzialisierung eignen, müssen diese eine entsprechend breite, bereits etablierte Entwicklergemeinde ansprechen können. Nur auf diese Weise können solide Projekte realisiert werden, die über gute Erfolgsaussichten verfügen und nur auf diese Weise ist dafür gesorgt, dass auch die folgenden Wartungsarbeiten ohne allzu hohe Kosten und Risiken durchgeführt werden können. Dafür müssen sich die aktuell vorhandenen RIA Frameworks noch weiter in die bereits vorhandenen, angenommenen Technologien integrieren und diese erweitern bzw. ergänzen. Wichtig ist dabei, dass diese RIA Frameworks nicht nur die Ergonomie der Applikation deutlich erhöhen, sondern dass sich diese neuen RIA Technologien ohne allzu viel Aufwand mit weiterhin schnellen Entwicklungszeiten in die vorhandene Softwareentwicklung einbetten lassen.

> *These* δ: Das technische erforderliche Wissen zur Umsetzung von RIA Anwendungen ist sehr umfangreich. Es ist davon auszugehen, dass sich langfristig Technologien durchsetzen werden, die sich in das vorhandene Enterprise Java Umfeld integrieren lassen (wie z.B. auch AJAX), weil hier bereits eine große Entwicklergemeinde vorhanden ist.

Zu den derzeit wichtigsten Kandidaten von RIA Technologien zählt AJAX mit den zugehörigen Tools und Frameworks, weil AJAX mit seinen Tools und Frameworks alle zuvor formulierten Voraussetzungen an eine durchsetzungsfähige RIA Technologie erfüllt. Man kann insbesondere davon ausgehen, dass AJAX von der aktuell vorhandenen Java Entwicklergemeinde bereits angenommen worden ist. Mit Hilfe von AJAX lassen sich RIA Anwendungen relativ zügig umsetzen. Weiterhin fügen sich AJAX Technologien sowohl technologisch als auch gesellschaftlich in die vorhandenen Technologielandschaften bzw. in die vorhandenen Entwicklergemeinden ein.

5.1.4 Browserabhängigkeit / Testing von RIA Anwendungen

> *These* ϵ: Mit wachsendem Umfang und Komplexität der Anforderungen an die Präsentationsschicht von Webanwendungen, sowie mit wachsendem Umfang und Komplexität der verfügbaren Technologien für diese Präsentationsschicht werden sich auch die zugehörigen Methoden und Vorgehensweisen dahingehend anpassen, dass beim Testen einer Webanwendung das Verhalten des Clients mit seinen statischen und dynamischen Eigenschaften stärker berücksichtigt werden wird, als es bisher bei den klassischen Webanwendungen der Fall war.

Im Gegensatz zu Rich Client Applikationen, bei denen das Client-System direkt installiert wird, sind RIAs abhängig von den Browsern. Diese Abhängigkeit von den Browsern stellt eine ernsthafte Schwierigkeit bei der Verwendung von RIAs dar. Im Intranet ist diese Hürde deutlich geringer als im Internet, da man im Intranet seine Rich Internet Application von vornherein nur für eine unternehmensweit eingeschränkte Menge von Browser-Produkten und -Versionen anbieten muss. Für das Internet ist die Schwierigkeit deutlich größer, eine RIA hinreichend unabhängig vom Browser zu bekommen. Was das Internet betrifft, sollte bereits beim Entwurf besonderes Augenmerk auf diese Schwierigkeit gelegt werden. Eine Möglichkeit ist hier immer, wenigstens eine Alternative über eine schnelle Weiterleitung auf eine *klassische* Seite anzubieten, wenn die RIA im aktuellen Browser nicht verwendet werden kann. Die Browserabhängigkeit von RIAs stellt eine spezielle Anforderung an das Testen von RIA Anwendungen dar, die es bei den klassischen Webanwendungen in dieser Form nicht gab und die entsprechend berücksichtigt werden muss. Es ist jedoch nicht nur die Browserabhängigkeit, die spezielle Anforderungen für das Testen darstellen. Es bestehen in dieser Hinsicht noch weitere, RIA-spezifische Herausforderungen, wie die *Back Button Problematik*. Bei dieser Besonderheit besteht das Problem, dass der Nutzer bei Benutzung des Zurück-Buttons möglicher Weise Aktionen, die im Hintergrund verlaufen sind rückgängig macht. Damit kann es leicht zu inkonsistenten Daten, sowohl auf dem Client, wie auf dem Server kommen.

Fasst man diese Herausforderungen an das Testen von RIA Anwendungen zusammen, so kann man feststellen, dass das Verhalten des Clients mit seinen Eigenschaften stärker berücksichtigt werden muss, als es bisher bei den klassischen Webanwendungen der Fall war.

5.1.5 Markt der RIA Produkte

> *These* ζ: RIA Technologien werden sich im Internet voraussichtlich dort etablieren, wo der Wettkampf der Anbieter um Kunden besonders hart ist, wie zum Beispiel bei Suchmaschinen, B2C Portalen oder wie z.B. bei B2B Marktplätzen. RIA Technologien werden sich im Intranet voraussichtlich dort etablieren, wo gute Software-Ergonomie zu Einsparungen führt, zum Beispiel im Zusammenhang mit einer komfortablen Leistungserfassung im Intranet.

Besonders zwei große Motive existieren, um auch in wirtschaftlich schwierigeren Zeiten in RIA Technologien zu investieren:

- Im Internet – das Erreichen von Attraktivität für den Benutzer und somit das Erringen von Vorteilen im Vergleich zum Mitbewerber. Verbesserungen eines Portals betreffen bei einem Portalbetreiber durchaus sein Kerngeschäft. Auf der Suche nach Kunden muss der Portalbetreiber die Benutzerfreundlichkeit seiner Page optimieren, um sich dadurch positiv von seinen Mitbewerbern zu unterscheiden. Die Ergonomie seiner Software gehört hier zu den erfolgskritischen Faktoren für sein Unternehmen.

- Im Intranet dreht es sich hauptsächlich darum, häufig verwendete Applikationen möglichst ergonomisch einer Masse von Mitarbeitern verfügbar zu machen. Beispielsweise eine Zeiterfassung, Mitarbeiter-Einsatzplanung, Kalender oder Mailing. Je höher die Bedienfreundlichkeit der dort zur Verfügung gestellten Applikationen ist, umso schneller können die Benutzer, also die Mitarbeiter des Unternehmens, mit diesen Applikationen arbeiten. Durch die Bedienfreundlichkeit der Intranet-Anwendungen wird also letztendlich Arbeitszeit eingespart.

Auf diese Weise können RIA Anwendungen im Intranet zu massiven Einsparungen bei den Gemeinkosten eines Unternehmens führen. Gerade hier liegt allerdings auch die Schwierigkeit beim Werben für den Einsatz von RIA Anwendungen: Der ROI (Return of Investment) von RIA Investitionen lässt sich schwer zuordnen: Weil eine Investition in RIA eben auf ein Absenken der Gemeinkosten abzielt, die sich in der Regel aber nicht unmittelbar einer Kostenstelle und damit einem Entscheidungsträger zuordnen lassen.

5.1.6 Modellierung

> *These* η: Mit wachsendem Anforderungen an die Präsentationsschicht, sowie dem steigendem Umfang und Komplexität der verfügbaren Technologien für diese Schicht, werden sich auch die zugehörigen Methoden und Vorgehensweisen dahingehend anpassen, dass bereits während der Designphase dem Entwurf des Clients mit seinen statischen und dynamischen Eigenschaften eine größere Bedeutung zukommt, als es bisher bei den klassischen Webanwendungen der Fall war.

Weil sich RIA Anwendungen in Hinsicht auf ihr Verhalten ähnlich komplex wie Rich Client Anwendungen verhalten, ist davon auszugehen, dass auch die Vorgehensweisen und Methoden bei der Entwicklung von RIA Anwendungen den Vorgehensweisen und Methoden bei der Entwicklung von Rich Client Applikationen ähnlicher werden. Zum Beispiel wird bei der Entwicklung einer RIA Anwendung bereits während der Designphase für die Präsentationsschicht dem Entwurf des Clients mit seinen statischen und dynamischen Eigenschaften eine größere Bedeutung zukommen, als es bei der Entwicklung einer klassischen Webanwendung der Fall gewesen wäre. Wenn man zum Beispiel bestimmte Daten beim erstmaligen Aufruf sofort synchron laden möchte, eine Menge von anderen Daten danach erst asynchron geladen werden sollen, dann macht es Sinn, sein Modell schon in dieser Hinsicht zu gestalten.

5.1.7 Barrierefreiheit

> *These* θ: Ein spezielles Thema stellt „Barrierefreiheit“ dar. Barrierefreie Unternehmen sind zugleich auch soziale Unternehmen der Zukunft. RIA wird voraussichtlich eine zentrale Stellung einnehmen bei der Umsetzung eines barrierefreien Intranets bzw. Internets.

Das Thema Barrierefreiheit wird auch in Zukunft weiter seine Bedeutung haben. Es ist zu erwarten, dass spezielle mobile Endgeräte den behinderten Mitbürgern einen verbesserten Zugang zum Internet/Intranet ermöglichen werden. Hier sind spezielle Technologien für die Präsentationsschicht von Webanwendungen gefragt, die von ihrer Art und Weise eine große Ähnlichkeit zu den vorhandenen RIA Technoligen aufweisen. Im Zusammenhang damit wird an dieser Stelle auch noch einmal ARIA erwähnt. Es ist daher davon auszugehen, dass RIA bei der Umsetzung eines barrierefreien Intranets bzw. Internets oder zumindest bei der Realisierung eines verbesserten Zugangs zum Internet/Intranet eine zentrale Stellung einnehmen wird.

5.2 Beispiel: Modellierung einer Präsentationsschicht

Im vorhergehenden Abschnitt wurden mit Hilfe von Thesen die Auswirkungen von RIA auf die Entwicklung vorgestellt und diskutiert. In diesem Abschnitt soll nun ein Beispiel dafür gegeben werden, wie sich ein Client-Design mit typischen RIA Verhalten auf die Modellierung des Fachdatenmodells, innerhalb der Präsentationsschicht des Servers, auswirken kann. Dieses Beispiel lässt sich der oben formulierten *These* η[1] zuordnen und soll die gestiegenen Anforderungen an den Client-Entwurf auf die serverseitige Modellierung dokumentieren und veranschaulichen.

Das Beispiel hat als Grundlage ein relativ umfangreiches Softwareprojekt der IBM in Deutschland. Bei diesem Projekt handelt es sich um ein Dokumenten-Management-System (DMS), bei dem mittelbar über eine Service-Sicht Dokumente in einem DMS-Backend gespeichert werden sollen.

Dem Benutzer wird dabei auf seinem Desktop ein so genannter *elektronischer Schreibtisch* angeboten. Dieser elektronische Schreibtisch besteht im wesentlichem aus einem Rich Client, der auf der linken Seite einen Navigationsbaum bietet. In der Mitte des Bildschirms bietet sich dem Benutzer eine Eingabemaske für die Erfassung von als *Metatdaten* bezeichneten Attributen des Dokuments. Auf einem zweiten Bildschirm präsentieren sich innerhalb eines PDF-Viewers die *Primärdaten* des Dokuments, also letztendlich das zuvor von einem Scan-Dienstleister eingescannte Dokument im PDF Format. Der Benutzer kann nun die Metadaten des Dokuments erfassen bzw. ändern, zum Dokument Notizen oder Entscheidungen erfassen, und das Dokument mit seinen erfassten bzw. geänderten Daten speichern.

Anzumerken ist, dass mit der sehr kurzen Erläuterung keine hinreichende Beschreibung der Systemfunktionalität gegeben worden ist. Es fehlt die hinreichend ausführliche Übersicht über die fachlichen Anforderungen, die Komponentenübersicht und ähnliche Beschreibungen. Ebenso fehlt die Beschreibungen für die Geschäftsprozesse und Anwendungsfälle der hier angedeuteten Software. Anstelle dessen wird auf einen bestimmten Aspekt dieser Software nun genauer eingegangen.

[1] Siehe 5.1.6

Es ist denkbar, eine Abwandlung dieser Software als RIA im Intranet zur Verfügung zu stellen. Auf diese Weise könnten Kunden mit gewissen Zugriffsrechten auf bestimmte Dokumente des DMS Backends über das Internet zugreifen.

Es soll nun betrachtet werden, wie das interne Modell der fachlichen Domäne angepasst werden müsste, damit es den Anforderungen einer Rich Internet Application genügt. Dafür muss zunächst das Modell der fachlichen Domäne der Dokumentenverwaltung vorgestellt werden. Die nachfolgende Abbildung zeigt das Modell der Komponente *Dokumentenverwaltung* innerhalb des DMS-Systems, also die fachlichen Entitäten der Dokumentenverwaltung, wie sie auf dem Server modelliert worden sind:

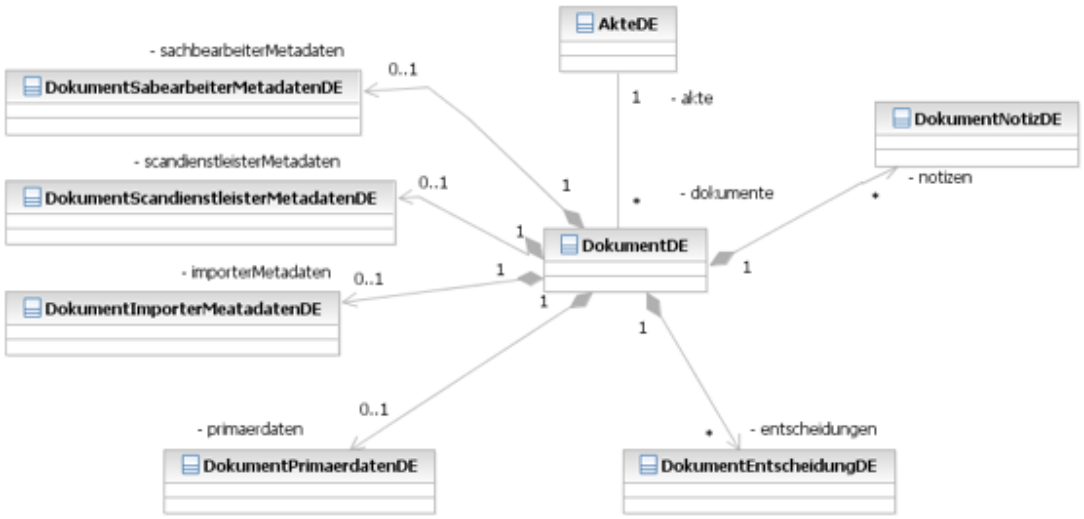

Abbildung 5.1: UML Diagramm zum Domain Modell der Komponente Dokumentenverwaltung des DMS Systems

Die Abkürzung *DE* als Extension des Klassennamens soll verdeutlichen, dass es sich bei der Klasse um einen Domain Entität handelt. Ein primäres, zentrales Objekt, dass als *Kopf* des Objektgeflechtes existiert und von dem aus die assoziierten bzw. die aggregierten sekundären Entitäten referenziert werden, soll hier als Modellierungsmuster dienen. Das *primäre* Objekt ist hier das DokumentDE. Fasst alle anderen Objekte sind sekundäre, aggregierte Objekte des Dokuments. Einzige Ausnahme bildet AkteDE, diese Entität ist eine primäre Entität aus der anderen Komponente *Aktenverwaltung* des DMS Systems. Bei den Aggregationen handelt es sich genau genommen um Kompositionen.

Die MetadatenKlassen sind Container, die zahlreiche Attribute als Eigenschaften des Dokumentes enthalten. Die Metadaten wurden dabei in diesem fiktiven Beispiel nach ihrer Herkunft unterteilt. Unterschieden wird dabei nach:

- Metadaten zum Dokument, die bereits beim Einscannen des Dokuments durch den Scandienstleister erstellt worden sind, falls dieses Dokument über die Poststelle und den nachgeschalteten Scandienstleister in das System gelangt ist,
- Metadaten zum Dokument, die beim Importieren des Dokuments aus einem anderen System erstellt worden sind, falls dieses Dokument importiert wurde und nach
- Metadaten zum Dokument, die ein Sachbearbeiter in Rahmen einer Fallbearbeitung zum eingegeben hat, falls das Dokument bereits von einem Sachbearbeiter bearbeitet wurde.

Die folgende Tabelle dokumentiert das oben präsentierte Modell.

Entität	**Beschreibung**
AkteDE	Zentrale Entität der Aktenverwaltung. Eine Akte kann mehrere Dokumente enthalten. Ein Dokument muss jedoch innerhalb von genau einer Akte sein.
DokumentDE	Zentrale Entität der Dokumentenverwaltung; besteht aus seiner ID, sowie aus diversen Beziehungen zu seinen Aggregationen
DokumentSachbearbeiter-MetadatenDE	Metadaten zum Dokument, die über den Sachbearbeiter in das System kamen
DokumentScandienstleister-MetadatenDE	Metadaten zum Dokument, die über den Scandienstleister in das System kamen
DokumentImporterMetadatenDE	Metadaten zum Dokument, die über eine ImporterSchnittstelle in das System kamen
DokumentPrimaerdatenDE	Primärdaten des Dokuments, in der Regel in Form einer PDF-Datei, (das Dokument selbst).
DokumentNotizDE	Notizen, die ein Sachbearbeiter im Rahmen einer Fallbearbeitung zu diesem Dokument hinterlegt hat.

Tabelle 5.1: Dokumentation zum Domain Modell der Komponente Dokumentenverwaltung des DMS Systems

Nun soll der interne Kern der Dokumentenverwaltung auf dem Server Funktionen anbieten, die es dieser RIA Präsentationsschicht ermöglichen, Anfragen eines RIA Clients zu bedienen. Unter anderem weil die Programmierung dieser Präsentationsschicht basierend auf AJAX Technologien für sich genommen ohnehin schon komplex genug ist, macht es Sinn, sich zeitnah über eine Aufbereitung der Daten für den Client auf dem Server Gedanken zu machen. Dafür wiederum muss der Entwurf des RIA Clients bekannt sein.

Wir gehen in diesem Beispiel davon aus, dass sich der RIA Client, nachdem er sich beim erstmaligen Aufruf der RIA DMS Anwendung initialisiert hat, auf Anfrage *ein Dokument holen* kann. *Ein Dokument holen* meint dabei, dass sich der Client bei der erstmaligen Anforderung nach dem Dokument zuerst einmal ganz bestimmte DokumentMetadaten anfordert, die sofort innerhalb von bestimmten Feldern angezeigt werden sollen. Im Anschluss fordert sich der Client asynchron eine Liste von weniger zentralen DokumentMetadaten an, die der Client dann innerhalb einer Liste in Form von (Schlüssel, Wert) Paaren dem Benutzer anzeigt. Es braucht eine Weile, bis sich diese Liste aufbaut. Aber da die Werte weniger zentral sind, wie die bereits beim synchronen Request angeforderten Werte und da der Prozess im Hintergrund geschieht, hat man an der Stelle keine kritische Einschränkung der User Experience. Für das Anfordern dieser Liste mit den zusätzlichen Metadaten soll ein AJAX/JavaScript Mechanismus verwendet werden. Das PDF zum Dokument soll beim Absetzen des erstmaligen Requests nach dem Dokument jedoch nicht angefordert werden. Anstelle dessen wird dem Benutzer auf dem Client ein Button *Download* angeboten, über den der Benutzer das PDF herunter laden kann. Weil für den Download des PDFs ein Streaming Mechanismus verwendet werden soll, kommt es nach Betätigen der Download-Schaltfläche zum Aufruf eines entsprechenden serverseitigen Servlets, über das das PDF zum Dokument auf den Client herunter geladen werden kann.

Für den Download der PDF Datei wird also keine *RIA Technologie*, sondern *nur* ein einfaches Servlet, verwendet. Man kann aber anhand dieses Beispiels bereits gut erkennen, wie RIA typische Technologien mit den bereits vorhandenen, im Web üblichen Technologien zusammen arbeitet um eine RIA Anwendung zu realisieren.

Um den oben angedeuteten RIA Client mit Daten geeignet versorgen zu können, macht der bisherige Entwurf für die Domäne *Dokumentverwaltung* keinen großen Sinn mehr, weil sich dort der Entwurf der Metadaten an der *Herkunft*

dieser Metadaten orientiert, und nicht daran, auf welche Weise und zu welchem Zeitpunkt diese Daten vom Client angefordert werden. Damit dieses auf übersichtliche Weise geschehen kann macht es wiederum Sinn, das fachlich aufbereitete Modell für die Präsentationsschicht auch wieder mittels UML zu entwerfen bzw. zu dokumentieren. Dazu folgt das UML Diagramm zu diesem Application Modell:

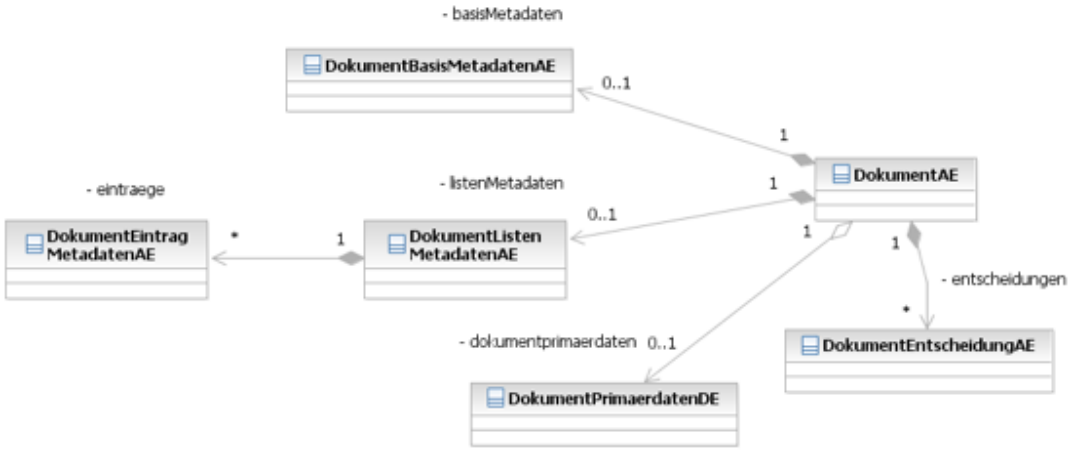

Abbildung 5.2: UML Diagramm zum Application Modell der Komponente Dokumentenverwaltung des DMS Systems

In diesem Modell werden nun die Metadaten des Dokuments also danach unterschieden, ob diese vom Client

- direkt beim erstmaligen Absetzen der Anfrage nach den Dokumentdaten angefordert werden,
- asynchron nach dem erstmaligen Absetzen der Anfrage nach den Dokumentdaten angefordert werden, oder
- nach Betätigen der „Download“ Schaltfläche auf dem Client über ein Servlet angefordert werden.

Die Metadaten des Domain Modells der Dokumentenverwaltung wurden also anstelle des Aspekts der fachlichen Herkunft der Daten nun neu, nach der clientseitigen Darstellung der Daten, sortiert.

Die nachfolgende Tabelle dokumentiert das oben präsentierte Modell:

Entität	**Beschreibung**
DokumentAE	Zentrale Entität der Dokumentenverwaltung für das Application Modell; besteht aus seiner ID, sowie aus diversen Beziehungen zu seinen Assoziationen bzw. Aggregationen.
DokumentBasisMetadatenDE	Metadaten zum Dokument, die sofort geladen und auf dem Client innerhalb von festen Feldern angezeigt werden sollen.
DokumentListenMetadatenDE	Verwalter für die Liste von Metadaten, die asynchron geladen werden und in Form einer Liste angezeigt werden sollen.
DokumentEintragMetadatenDE	Metadaten Listeneintrag (Schlüssel, Wert) zum Dokument, der im Client innerhalb der asynchron geladenen Liste angezeigt werden soll.
DokumentPrimaerdatenDE	Primärdaten des Dokuments, in der Regel in Form einer PDF-Datei, praktisch also das Dokument selber.
DokumentEntscheidungAE	Die Daten der Entscheidung(en), die ein Sachbearbeiter im Rahmen einer Fallbearbeitung zu diesem Dokument getroffen hat.

Tabelle 5.2: Dokumentation zum Application Modell der Komponente Dokumentenverwaltung des DMS Systems

Bemerkung: Im Falle der Entität *DokumentPrimaerdatenDE* muss es sich nicht um das tatsächliche Dokument, sondern wie zuvor erwähnt kann es sich dabei um eine Referenz auf die PDF-Datei handeln, welche über ein Servlet auf den Client übertragen wird. Denkbar wäre es jedoch, die einmal geladenen Dokumente nicht nur flüchtig, sondern persistent auf der Seite des Client zu speichern. Lohnen würde sich dies jedoch nur bei einer häufigen Verwendung einzelner Dokumente. Inwiefern also eine dauerhafte Speicherung einzelner Dokumente sinnvoll ist, hängt vom Einzelfall ab. Daher wäre eine Abfrage beim Verlassen der Applikation denkbar, ob die Dokumente dieser Sitzung, auch noch in der Nächsten zur Verfügung stehen sollen.

Eine solche Form der Aufbereitung von Daten für das aufrufende ClientSystem sowie die zugehörige Art der Datenmodellierung ist für RichClient Systeme

bereits üblich. Neu hier ist jedoch, dass diese Prinzipien nun zusammen mit den sich verbreitenden RIA Technologien auch in die Entwicklung von Web-Anwendungen Einzug halten, um den wachsenden Anforderungen der Kunden an die Ergonomie dieser Art von Applikationen genügen zu können.

Dieses Beispiel zur Modellierung einer Präsentationsschicht bekräftigt die *These* η, nach welcher bereits in die Modellierung dieser Schicht, dem Entwurf des Clients größere Bedeutung zukommt. Denn das Domain Modell wurde nicht nach der Herkunft der Daten, sondern nach der Verwendung bzw. Darstellung im Client erstellt. Die Gründe dafür liegen in den Anforderungen die RIAs an die Präsentationsschicht stellen, bei welchem die Software Ergonomie bzw. die User Experience einen wesentlichen Bestandteil ausmacht.

6 Zusammenfassung

Das letzte Kapitel soll mit einem Fazit zu den Auswirkungen auf die Entwicklung und einem Ausblick, für sich anschließende Themen, die Arbeit abrunden. Das Fazit soll dabei ein Zusammenfassung aller gewonnenen Erkenntnisse sein.

6.1 Fazit

Zum Abschluss dieser Arbeit sollen die erzielten Erkenntnisse und Resultate in kurzer Form zusammengefasst werde:

- Die Software Ergonomie, sowie die daraus resultierende User Experience und damit das User Interface sind - gerade im Zeitalter von Web 2.0 - Schlüsselaspekte von Rich Internet Applications.
- Resultierend aus den hohen Ansprüchen der Software Ergonomie, muss bereits bei der Modellierung der Software, den spezifischen Merkmalen von Rich Internet Applications besondere Aufmerksamkeit zuteil werden, da diese die Ergonomie wesentlich beeinflussen.
- Ein fundiertes Fachwissen über Rich Internet Applications, ist demnach ebenso Grundvorraussetzung für die erfolgreiche Entwicklung von RIAs, wie das umfangreiche Wissen über die spezielle RIA-Technologie, mit welcher diese Software realisiert werden soll.

- Durch die hohen Anforderungen an die Benutzerfreundlichkeit, kommt es unweigerlich zu einem starken Anstieg der Komplexität der Software. Daraus resultieren steigenden Entwicklungszeiten und Entwicklungskosten, die durch die Verwendung moderner Frameworks oder Toolkits nur zu einem gewissen Teil ausgeglichen werden können.

- Die stark angestiegene Komplexität führt, ebenso wie die Verteilung von Daten und Geschäftslogik, zu erheblichen Schwierigkeiten beim Testen der Anwendung. Vor allem umfangreiche Tests der interaktiven Präsentationsschicht und die Minimierung von Browserinkompatibilität sind für die Qualitätssicherung der Applikation ausschlaggebend.

- Ein besonderes Thema stellt die Barrierefreiheit dar, denn die zunehmende Anzahl an entsprechend leistungsfähigen mobilen Endgeräten, bietet neue Möglichkeiten für die Entwicklung verteilter Software. Jedoch treten durch verschiedene Geräte und Hersteller zahlreiche Probleme auf. So ist neben der Browserinkompatibilität in Zukunft auch die Gerätekompatibilität ein nicht zu vernachlässigender Faktor. Eine Standardisierung von Software und Schnittstellen ist die wichtige Vorraussetzung für das Lösen dieser Problematik.

Rich Internet Applications haben demnach, durch ihre spezifischen Merkmalen, umfangreiche Auswirkungen auf die Entwicklung von Informationssystemen. Die Ursache dieser Auswirkungen sind dabei nahezu ausschließlich in der Steigung der Komplexität, durch die erhöhten Anforderungen der Software Ergonomie, zu suchen. Entgegenwirken lassen diese sich nur mit umfangreichen Wissen und der entsprechenden Umsichtigkeit beim Entwicklungsprozess, von der Modellierung bis hin zum finalen Test der Anwendung.

6.2 Ausblick

Die Arbeit setzt sich allgemein mit den spezifischen Merkmalen von Rich Internet Applications, sowie den Auswirkungen auf die Entwicklung auseinander und vermittelt draus folgend vorrangig Überblickswissen. Interessant wäre es jedoch ebenfalls, sich mit einzelnen, angesprochenen Themen intensiver zu befassen. So wäre es denkbar das Thema *Barrierefreiheit* im Zusammenhang mit Web 2.0 und semantischen Web zu betrachten und dazu die konkrete praktische Umsetzung verschiedener RIA-Technologien zu vergleichen. Da die Barrierefreiheit, nicht nur aus sozialer Sichtweise, in Zukunft sicher eine verstärkte Rolle spielen wird.

Teil II

Anhang

A
Abkürzungsverzeichnis

Kürzel	Beschreibung	
AMF	Action Message Format	41
ARIA	Accessible Rich Internet Applications	31
B2B	Business to Business	48
B2C	Business to Consumer	48
CERN	Conseil Européen pour la Recherche Nucléaire	5
CSS	Cascading Style Sheets	29
DBMS	Datenbank Management System	11
DMS	Dokumenten-Management-System	54
DOM	Document Objekt Model	34
ERM	Entity-Relationship-Modell	20
GWT	Google Web Toolkit	37
JSON	JavaScript Object Notation	43
JVM	Java Virtual Machine	29
REST	Representational State Transfer	44
RIA	Rich Internet Application	1
RTMP	Real-Time Messaging Protokoll	27
SOAP	Simple Object Access Protocol	8
SWF	Shockwave Flash	27

Kürzel	**Beschreibung**	
UI	User Interface	22
UML	Unified Modeling Language	20
UX	User Experience	33
W3C	World Wide Web Consortiums	5
WPF	Windows Presentation Framework	44
WWW	World Wide Web	7
XAML	Exensible Application Markup Language	44
XML	Extensible Markup Language	11
XUL	XML User Interface Language	26, 29

B.1 Quellen Kapitel 2

B.1.1 Internetquellen

[URL01] Bericht Berners-Lee
`http://www.w3.org/History/1989/proposal.html`
verfügbar am 10.08.2009

[URL02] Browser Viola
`http://www.xcf.berkeley.edu/~wei/`
verfügbar am 10.08.2009

[URL03] WWW History
`http://www.w3.org/History.html`
verfügbar am 10.08.2009

[URL04] W3C Timeline
`http://www.w3.org/2005/01/timelines/timeline-2500x998.png`
verfügbar am 10.08.2009

[URL05] Tim O'Reilly: What is Web 2.0
`http://oreilly.com/web2/archive/what-is-web-20.html`
verfügbar am 10.08.2009

[URL06] Bob Baxley: What is a Web Application?
`http://www.boxesandarrows.com/view/what_is_a_web_application_`
verfügbar am 11.08.2009

[URL07] Rakesh Pai: Web applications- The Wave of Future
http://piecesofrakesh.blogspot.com/2005/01/web-applications-wave-of-future.html
verfügbar am 11.08.2009

[URL08] O'Callahan: Essence of Web Applications
http://weblogs.mozillazine.org/roc/archives/2008/11/the_essence_of.html
verfügbar am 11.08.2009

[URL09] Paul Graham: The Other Road Ahead
http://www.paulgraham.com/road.html
verfügbar am 12.08.2009

[URL10] John Gruber: The Location Field Is the New Command Line
http://daringfireball.net/2004/06/location_field
verfügbar am 12.08.2009

[URL11] Robert Chartier: Application Architecture: An N-Tier Approach
http://www.15seconds.com/issue/011023.htm
verfügbar am 12.08.2009

[URL12] n-Tier/n-Layer
http://davidhayden.com/blog/dave/archive/2005/07/22/2401.aspx
verfügbar am 12.08.2009

[URL13] Jeremy Petersen: Benefits of using the n-tier approach for web applications
http://www.adobe.com/devnet/coldfusion/articles/ntier.html
verfügbar am 12.08.2009

B.2 Quellen Kapitel 3

B.2.1 Literatur

[LIT01] Dana, Moore; Raymond, Budd; Edward, Benson: Professional Rich Internet Applications: AJAX and Beyond. - 1.Aufl. - Indianapolis, Indiana: Wiley Publishing, Inc., 2007

[LIT02] Lars Röwekamp: Im üblichen Rahmen in: iX Special Web on Rails. - Augabe 2/2009 Juni,Juli,August

[LIT03] Preciado, J.C.; Linaje, M; Comai, S...: Designing Rich Internet Applications with Web Engineering Methodologies. -1.Aufl. - Quercus Software Engineering group. Universidad de Extremadura (Cáceres, Spain) Politecnico di Milano. Dipart. di Elettronica e Informazione (Milano, Italy), 2007

[LIT04] Leff, A.; Rayfleld, J.: Programming model alternatives for disconnected business applications, Internet Computting Magazine, IEEE press, Los Alamitos (USA), 2006

[LIT05] Bozzon A.; Comai S.; Fraternali P. ...: Conceptual Modeling and Code Generation for Rich Internet Applications, Springer, California (USA), 2006

[LIT06] Thomas Walter : Kompendium der Webprogrammierung. -1.Aufl. - Springer, Berlin, 2007

[LIT07] Zetie, Carl: The Rise Of Rich Internet Applications. Forrester Research Inc., 2006 - Forschungsbericht

B.2.2 Internetquellen

[URL14] RSS Wikipedia
http://de.wikipedia.org/wiki/RSS
verfügbar am 20.07.2009

[URL15] Joshua, Duhl: Rich Internet Applications. Framingham USA: IDC, 2003
http://www.adobe.com/platform/whitepapers/idc_impact_of_rias.pdf
verfügbar am 20.07.2009

[URL16] Chris Loosley: Rich Internet Applications: Design, Measurement, and Management Challenges. SLM Technologies Keynote Systems, 2006
http://keynote.com/docs/whitepapers/RichInternet_5.pdf
verfügbar am 20.07.2009

[URL17] Wikipedia Flash
http://de.wikipedia.org/wiki/Adobe_Flash
verfügbar am 28.07.2009

[URL18] RTMP Spezifikation
http://www.adobe.com/devnet/rtmp/pdf/rtmp_specification_1.0.pdf
verfügbar am 29.07.2009

[URL19] Jesse James Garett, Ajax: A New Approach to Web Applications
http://adaptivepath.com/ideas/essays/archives/000385.php
verfügbar am 29.07.2009

[URL20]	Barrierefreiheit http://de.wikipedia.org/wiki/Barrierefreies_Internet verfügbar am 01.08.2009
[URL21]	IWP 8/2005 Barrierefreiheit im Internet http://www.bit-informationsdesign.de/iwp-8-2005/index.html verfügbar am 01.08.2009
[URL22]	Brigitte, Bornemann-Jeske: Barrierefreies Webdesign zwischen Webstandards und Universellem Design - Hamburg, 2005 http://www.bit-informationsdesign.de/iwp-8-2005/IWP-8-2005-Bornemann-Jeske.pdf verfügbar am 01.08.2009
[URL23]	ARIA- Accessible Rich Internet Applications http://de.wikipedia.org/wiki/Accessible_Rich_Internet_Applications verfügbar am 01.08.2009
[URL24]	H. Loranger, A Schade, J. Nielsen; Usability of Rich Internet Applications and Web-Based Tools - USA 2002 http://www.nngroup.com/reports/flash/RIA-usability.pdf verfügbar am 03.08.2009
[URL25]	Donald Norman UX http://uxzentrisch.de/donald-norman-definiert-user-experience/ verfügbar am 01.08.2009
[URL26]	Interview http://www.adaptivepath.com/media/ap-interview-don_norman-peterme.mp3 verfügbar am 01.08.2009

B.3 Quellen Kapitel 4

B.3.1 Literatur

[LIT08]	Wang, Dapeng: Nicht nur holländische Fussballkunst: Java Magazin. - Ausgabe 11.05
[LIT09]	P.J. Deitel, H.M. Deitel: AJAX, Rich Internet Application and Web Development for Programmers. - 1 Auflage, Pearson Education Inc., Crawfordsville, Indiana, 2008
[LIT10]	F. Kretzschmar, I. Reiztenstein, M. Schramm; Ajax - Technologien in Model-2-Webanwendungen: Java mit Integrationsspektrum. - Ausgabe Februar/März 2007
[LIT11]	Steven Douglas Olson: Ajax und Java. – 1. Auflage O'Reilly 2007
[LIT12]	Michael Seemann: Das Google Web Toolkit GTW. – 1.Auflage O'Reilly 2007
[LIT13]	Simon Widjaja: Rich Internet Applications mit Adobe Flex 3. – 1.Auflage Hanser, 2008
[LIT14]	Ralph Steyer: JavaFX. Dynamische und interaktive Java-Applikationen mit Java FX. – 1.Auflage, Addison-Wesley, München; 2007
[LIT15]	G. Anderson, P. Anderson: Essential JavaFX. – 1. Auflage, Prentice Hall PTR, Crawfordsville, Indiana, 2009

[LIT16] Regina Dowling, Jörg Müller: Es werde Licht in: iX Special Web on Rails. - Augabe 2/2009 Juni,Juli,August

B.3.2 Internetquellen

[URL27] Matthew Russell: The Mojo of Dojo. 2007
http://onlamp.com/pub/a/onlamp/2007/11/01/the-mojo-of-dojo.html?CMP=AFC-ak_article&ATT=The+Mojo+of+Dojo
verfügbar am 05.08.2009

[URL28] DojoStorage
http://www.dojotoolkit.org/book/dojo-book-0-4/part-7-utilities/storage
verfügbar am 05.08.2009

[URL29] OpenLaszlo Performance Optimization
http://wiki.openlaszlo.org/Performance_Optimization
verfügbar am 05.08.2009

[URL30] OpenLaszlo Performance Monitoring and Tuning
http://www.openlaszlo.org/lps3/docs/guide/performance-tuning.html
verfügbar am 05.08.2009

C
Erklärung zur selbständigen Anfertigung

Erklärung:

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Bearbeitungsort, Datum Unterschrift